DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models

DeepSeek-AI

research@deepseek.com

Abstract

We introduce DeepSeek-V3.2, a model that harmonizes high computational efficiency with superior reasoning and agent performance. The key technical breakthroughs of DeepSeek-V3.2 are as follows: **(1) DeepSeek Sparse Attention (DSA)**: We introduce DSA, an efficient attention mechanism that substantially reduces computational complexity while preserving model performance in long-context scenarios. **(2) Scalable Reinforcement Learning Framework**: By implementing a robust reinforcement learning protocol and scaling post-training compute, DeepSeek-V3.2 performs comparably to GPT-5. Notably, our high-compute variant, DeepSeek-V3.2-Speciale, surpasses GPT-5 and exhibits reasoning proficiency on par with Gemini-3.0-Pro, achieving gold-medal performance in both the 2025 International Mathematical Olympiad (IMO) and the International Olympiad in Informatics (IOI). **(3) Large-Scale Agentic Task Synthesis Pipeline**: To integrate reasoning into tool-use scenarios, we developed a novel synthesis pipeline that systematically generates training data at scale. This methodology facilitates scalable agentic post-training, yielding substantial improvements in generalization and instruction-following robustness within complex, interactive environments.

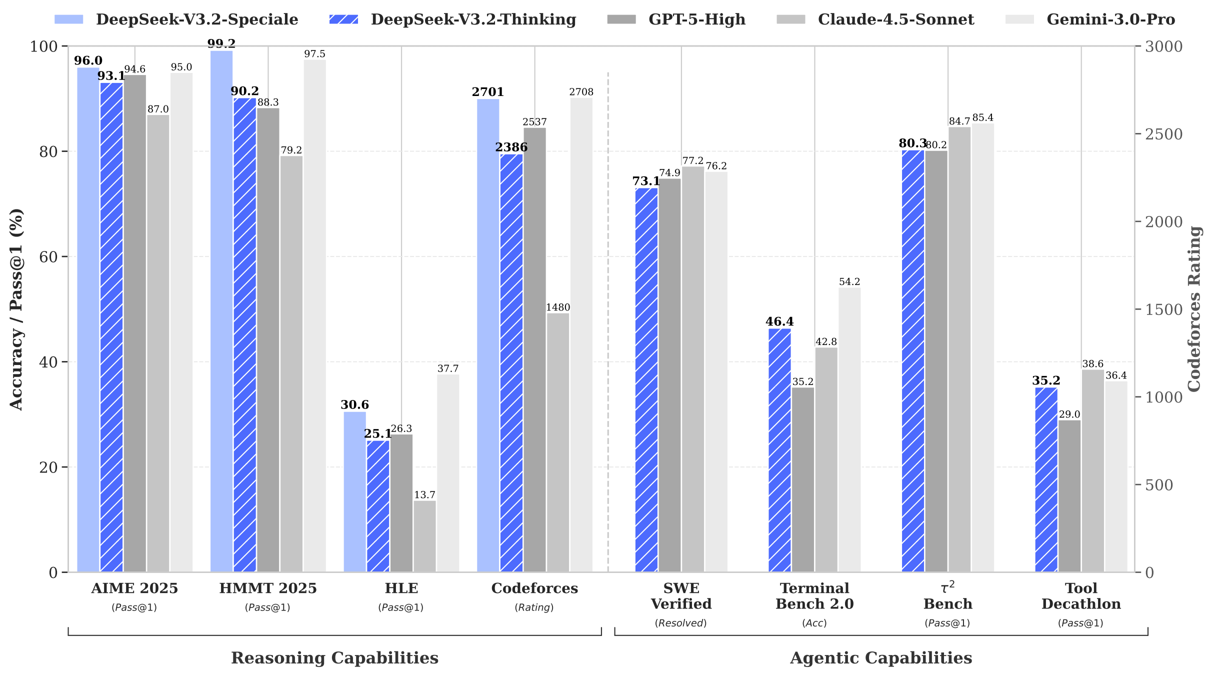


Figure 1 | Benchmark of DeepSeek-V3.2 and its counterparts. For HMMT 2025, we report the February competition, consistent with the baselines. For HLE, we report the text-only subset.

DeepSeek-V3.2：推动开放大型语言模型的前沿

DeepSeek-ai

`research@deepseek.com`

抽象的

我们推出了 DeepSeek-V3.2，这是一种将高计算效率与卓越推理和代理性能相结合的模型。 DeepSeek-V3.2的关键技术突破如下：（1）DeepSeek稀疏注意力（DSA）：我们引入了DSA，一种高效的注意力机制，可以大大降低计算复杂度，同时在长上下文场景中保持模型性能。 (2) 可扩展的强化学习框架：通过实施强大的强化学习协议和扩展训练后计算，DeepSeek-V3.2 的性能与 GPT-5 相当。值得注意的是，我们的高计算变体 DeepSeek-V3.2-Speciale 超越了 GPT-5，并表现出与 Gemini-3.0-Pro 相当的推理能力，在 2025 年国际数学奥林匹克（IMO）和国际信息学奥林匹克（IOI）中均获得金牌。 (3) 大规模代理任务合成管道：为了将推理集成到工具使用场景中，我们开发了一种新颖的合成管道，可以系统地大规模生成训练数据。这种方法有利于可扩展的代理后训练，在复杂的交互式环境中显着提高泛化性和指令遵循的鲁棒性。

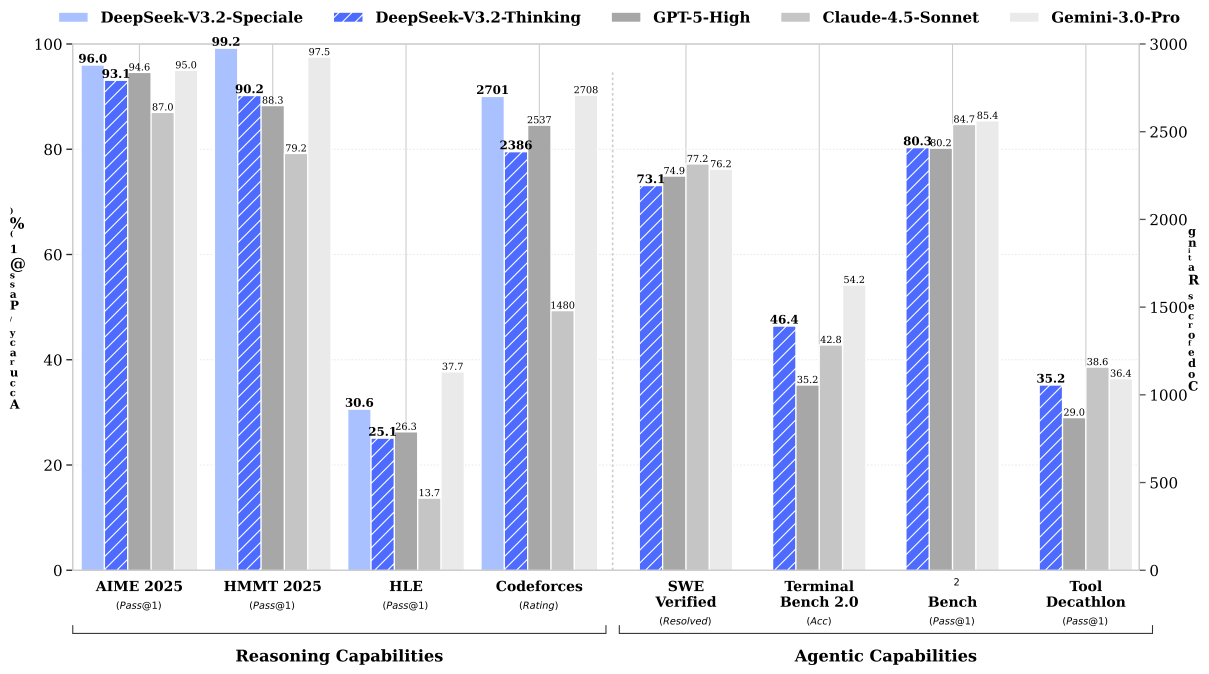


图 1 DeepSeek-V3.2 及其对应版本的 | 基准测试。对于 HMMT 2025，我们报告了 2 月份的比赛，与基线一致。对于 HLE，我们报告纯文本子集。

1. Introduction

The release of reasoning models (DeepSeek-AI, 2025; OpenAI, 2024a) marked a pivotal moment in the evolution of Large Language Models (LLMs), catalyzing a substantial leap in overall performance across the verifiable fields. Since this milestone, the capabilities of LLMs have advanced rapidly. However, a distinct divergence has emerged in the past months. While the open-source community (MiniMax, 2025; MoonShot, 2025; ZhiPu-AI, 2025) continues to make strides, the performance trajectory of closed-source proprietary models (Anthropic, 2025b; DeepMind, 2025a; OpenAI, 2025) has accelerated at a significantly steeper rate. Consequently, rather than converging, the performance gap between closed-source and open-source models appears to be widening, with proprietary systems demonstrating increasingly superior capabilities in complex tasks.

Through our analysis, we identify three critical deficiencies that limit the capability of open-source models in complex tasks. First, architecturally, the predominant reliance on vanilla attention (Vaswani et al., 2017) mechanisms severely constrains efficiency for long sequences. This inefficiency poses a substantial obstacle to both scalable deployment and effective post-training. Second, regarding resource allocation, open-source models suffer from insufficient computational investment during the post-training phase, limiting their performance on hard tasks. Finally, in the context of AI agents, open-source models demonstrate a marked lag in generalization and instruction-following capabilities compared to their proprietary counterparts (EvalSys, 2025; Li et al., 2025; Luo et al., 2025), hindering their effectiveness in real deployment.

To address these critical limitations, we first introduce DSA, a highly efficient attention mechanism designed to substantially reduce computational complexity. This architecture effectively addresses the efficiency bottleneck, preserving model performance even in long-context scenarios. Second, we develop a stable and scalable RL protocol that allows for significant computational expansion during the post-training phase. Notably, this framework allocates a post-training computational budget exceeding 10% of the pre-training cost, unlocking advanced capabilities. Thirdly, we propose a novel pipeline to foster generalizable reasoning in tool-use scenarios. First, we implement a cold-start phase utilizing the DeepSeek-V3 (DeepSeek-AI, 2024) methodology to unify reasoning and tool-use within single trajectories. Subsequently, we advance to large-scale agentic task synthesis, where we generate over 1,800 distinct environments and 85,000 complex prompts. This extensive synthesized data drives the RL process, significantly enhancing the model's generalization and instruction-following capability in the agent context.

DeepSeek-V3.2 achieves similar performance with Kimi-k2-thinking and GPT-5 across multiple reasoning benchmarks. Furthermore, DeepSeek-V3.2 significantly advances the agentic capabilities of open models, demonstrating exceptional proficiency on the long-tail agent tasks introduced in EvalSys (2025); Li et al. (2025); Luo et al. (2025). DeepSeek-V3.2 emerges as a highly cost-efficient alternative in agent scenarios, significantly narrowing the performance gap between open and frontier proprietary models while incurring substantially lower costs. Notably, with the aim of pushing the boundaries of open models in the reasoning domain, we relaxed the length constraints to develop DeepSeek-V3.2-Speciale. As a result, DeepSeek-V3.2-Speciale achieves performance parity with the leading closed-source system, Gemini-3.0-Pro (DeepMind, 2025b). It shows gold-medal performance in the IOI 2025, ICPC World Final 2025, IMO 2025, and CMO 2025.

一、简介

推理模型（DeepSeek-AI，2025；OpenAI，2024a）的发布标志着大型语言模型（LLM）发展的关键时刻，促进了可验证领域整体性能的大幅飞跃。自这一里程碑以来，法学硕士的能力迅速提高。然而，过去几个月出现了明显的分歧。尽管开源社区（MiniMax，2025；MoonShot，2025；ZhiPu-AI，2025）继续取得长足进步，闭源专有模型（Anthropic，2025b；DeepMind，2025a；OpenAI，2025）的性能轨迹却以更快的速度加速。因此，闭源模型和开源模型之间的性能差距似乎正在扩大，而不是趋同，专有系统在复杂任务中表现出越来越优越的能力。

通过分析，我们发现了限制开源模型在复杂任务中的能力的三个关键缺陷。首先，从架构上来说，对普通注意力（Vaswani et al., 2017）机制的主要依赖严重限制了长序列的效率。这种低效率对可扩展部署和有效的后期培训构成了重大障碍。其次，在资源分配方面，开源模型在训练后阶段的计算投入不足，限制了其在困难任务上的性能。最后，在人工智能代理的背景下，与专有模型相比，开源模型在泛化和指令跟踪能力方面表现出明显的滞后（EvalSys，2025；Li et al.，2025；Luo et al.，2025），阻碍了它们在实际部署中的有效性。

为了解决这些关键限制，我们首先引入 DSA，这是一种高效的注意力机制，旨在大幅降低计算复杂性。该架构有效解决了效率瓶颈，即使在长上下文场景下也能保持模型性能。其次，我们开发了一个稳定且可扩展的强化学习协议，允许在训练后阶段进行显着的计算扩展。值得注意的是，该框架分配的训练后计算预算超过训练前成本的 10%，从而释放了高级功能。第三，我们提出了一种新颖的管道来促进工具使用场景中的普遍推理。首先，我们利用 DeepSeek-V3（DeepSeek-AI，2024）方法实现冷启动阶段，以统一单个轨迹内的推理和工具使用。随后，我们进入大规模代理任务合成，生成超过 1,800 个不同的环境和 85,000 个复杂的提示。这些广泛的合成数据驱动强化学习过程，显着增强模型在代理环境中的泛化和指令跟踪能力。

DeepSeek-V3.2 在多个推理基准测试中实现了与 Kimi-k2-thinking 和 GPT-5 相似的性能。此外，DeepSeek-V3.2 显着提升了开放模型的代理能力，展示了对 EvalSys (2025) 中引入的长尾代理任务的卓越熟练程度；李等人。（2025）；罗等人。（2025）。DeepSeek-V3.2作为代理场景中极具成本效益的替代方案而出现，显着缩小了开放模型与前沿专有模型之间的性能差距，同时大幅降低了成本。值得注意的是，为了突破推理领域开放模型的界限，我们放宽了长度限制来开发 DeepSeek-V3.2-Speciale。因此，DeepSeek-V3.2-Speciale 的性能与领先的闭源系统 Gemini-3.0-Pro（DeepMind，2025b）相当。它在 IOI 2025、ICPC World Final 2025、IMO 2025 和 CMO 2025 中展现了金牌表现。

2. DeepSeek-V3.2 Architecture

2.1. DeepSeek Sparse Attention

DeepSeek-V3.2 uses exactly the same architecture as DeepSeek-V3.2-Exp. Compared with DeepSeek-V3.1-Terminus, the last version of DeepSeek-V3.1, the only architectural modification of DeepSeek-V3.2 is the introduction of DeepSeek Sparse Attention (DSA) through continued training.

Prototype of DSA. The prototype of DSA primarily consists of two components: a lightning indexer and a fine-grained token selection mechanism.

The **lightning indexer** computes the index score $I_{t,s}$ between the query token $\mathbf{h}_t \in \mathbb{R}^d$ and a preceding token $\mathbf{h}_s \in \mathbb{R}^d$, determining which tokens to be selected by the query token:

$$I_{t,s} = \sum_{j=1}^{H^I} w_{t,j}^I \cdot \mathrm{ReLU}\left(\mathbf{q}_{t,j}^I \cdot \mathbf{k}_s^I\right), \tag{1}$$

where H^I denotes the number of indexer heads; $\mathbf{q}_{t,j}^I \in \mathbb{R}^{d^I}$ and $w_{t,j}^I \in \mathbb{R}$ are derived from the query token $\mathbf{h}_t$; and $\mathbf{k}_s^I \in \mathbb{R}^{d^I}$ is derived from the preceding token $\mathbf{h}_s$. We choose ReLU as the activation function for throughput consideration. Given that the lightning indexer has a small number of heads and can be implemented in FP8, its computational efficiency is remarkable.

Given the index scores $\{I_{t,s}\}$ for each query token $\mathbf{h}_t$, our **fine-grained token selection mechanism** retrieves only the key-value entries $\{\mathbf{c}_s\}$ corresponding to the top-k index scores. Then, the attention output $\mathbf{u}_t$ is computed by applying the attention mechanism between the query token $\mathbf{h}_t$ and the sparsely selected key-value entries $\{\mathbf{c}_s\}$:

$$\mathbf{u}_t = \mathrm{Attn}\big(\mathbf{h}_t, \big\{\mathbf{c}_s \,\big|\, I_{t,s} \in \mathrm{Top\text{-}k}(I_{t,:})\big\}\big). \tag{2}$$

Instantiate DSA Under MLA. For the consideration of continued training from DeepSeek-V3.1-Terminus, we instantiate DSA based on MLA (DeepSeek-AI, 2024) for DeepSeek-V3.2. At the kernel level, each key-value entry must be shared across multiple queries for computational efficiency (Yuan et al., 2025). Therefore, we implement DSA based on the MQA (Shazeer, 2019) mode of MLA[1], where each latent vector (the key-value entry of MLA) will be shared across all query heads of the query token. The DSA architecture based on MLA is illustrated in Figure 2. We also provide an open-source implementation of DeepSeek-V3.2[2] to specify the details unambiguously.

2.1.1. *Continued Pre-Training*

Starting from a base checkpoint of DeepSeek-V3.1-Terminus, whose context length has been extended to 128K, we perform continued pre-training followed by post-training to create DeepSeek-V3.2.

The continued pre-training of DeepSeek-V3.2 consists of two training stages. For both stages, the distribution of training data is totally aligned with the 128K long context extension data used for DeepSeek-V3.1-Terminus.

[1]We illustrate the difference between the MQA and MHA modes of MLA in Appendix A.
[2]`https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp/tree/main/inference`

2. DeepSeek-V3.2架构

2.1. DeepSeek 稀疏注意力

DeepSeek-V3.2 使用与 DeepSeek-V3.2-Exp 完全相同的架构。与 DeepSeek-V3.1 的最后一个版本 DeepSeek-V3.1-Terminus 相比，DeepSeek-V3.2 唯一的架构修改是通过持续训练引入 DeepSeek 稀疏注意力（DSA）。

DSA的原型。 DSA的原型主要由两个组成部分组成：闪电索引和细粒的代币选择机制。

闪电索引器计算查询令牌 $\mathrm{h}_t \in \mathbb{R}^d$ 和前面的令牌 $\mathrm{h}_s \in \mathbb{R}^d$ 之间的索引分数 $I_{t,s}$，确定查询令牌要选择哪些令牌：

$$I_{t,s} = \sum_{j=1}^{H^I} w_{t,j}^I \cdot \mathrm{ReLU}\left(\mathbf{q}_{t,j}^I \cdot \mathbf{k}_s^I\right), \tag{1}$$

其中H^I表示索引器头的数量； $\mathrm{q}_{t,j}^I \in \mathbb{R}^{d^I}$ 和 $w_{t,j}^I \in \mathbb{R}$ 源自查询标记 h_t； $\mathrm{k}_s^I \in \mathbb{R}^{d^I}$ 是从前面的标记 h_s 导出的。出于吞吐量考虑，我们选择ReLU作为激活函数。鉴于闪电索引器的头数较少，并且可以在 FP8 中实现，其计算效率非常显着。

给定每个查询标记 h_t 的索引分数 $\{I_{t,s}\}$，我们的细粒度标记选择机制仅检索与前 k 个索引分数对应的键值条目 $\{\mathrm{c}_s\}$。然后，通过在查询标记 h_t 和稀疏选择的键值条目 $\{\mathrm{c}_s\}$ 之间应用注意机制来计算注意输出 u_t：

$$\mathbf{u}_t = \mathrm{Attn}\left(\mathbf{h}_t, \left\{\mathbf{c}_s \,\middle|\, I_{t,s} \in \mathrm{Top\text{-}k}\left(I_{t,:}\right)\right\}\right). \tag{2}$$

在 MLA 下实例化 DSA。出于对DeepSeek-V3.1-Terminus继续训练的考虑，我们为DeepSeek-V3.2实例化了基于MLA（DeepSeek-AI，2024）的DSA。在内核级别，每个键值条目必须在多个查询之间共享，以提高计算效率（Yuan et al., 2025）。因此，我们基于 MLA[1] 的 MQA (Shazeer, 2019) 模式实现 DSA，其中每个潜在向量（MLA 的键值条目）将在查询令牌的所有查询头之间共享。基于 MLA 的 DSA 架构如图 2 所示。我们还提供 DeepSeek-V3.2[2] 的开源实现，以明确指定细节。

2.1.1. Continued Pre-Training

从 DeepSeek-V3.1-Terminus 的基础检查点开始，其上下文长度已扩展至 128K，我们执行持续的预训练，然后进行后训练以创建 DeepSeek-V3.2。

DeepSeek-V3.2的持续预训练包括两个训练阶段。对于这两个阶段，训练数据的分布与 DeepSeek-V3.1-Terminus 使用的 128K 长上下文扩展数据完全一致。

[1]We illustrate the difference between the MQA and MHA modes of MLA in Appendix A.
[2]`https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp/tree/main/inference`

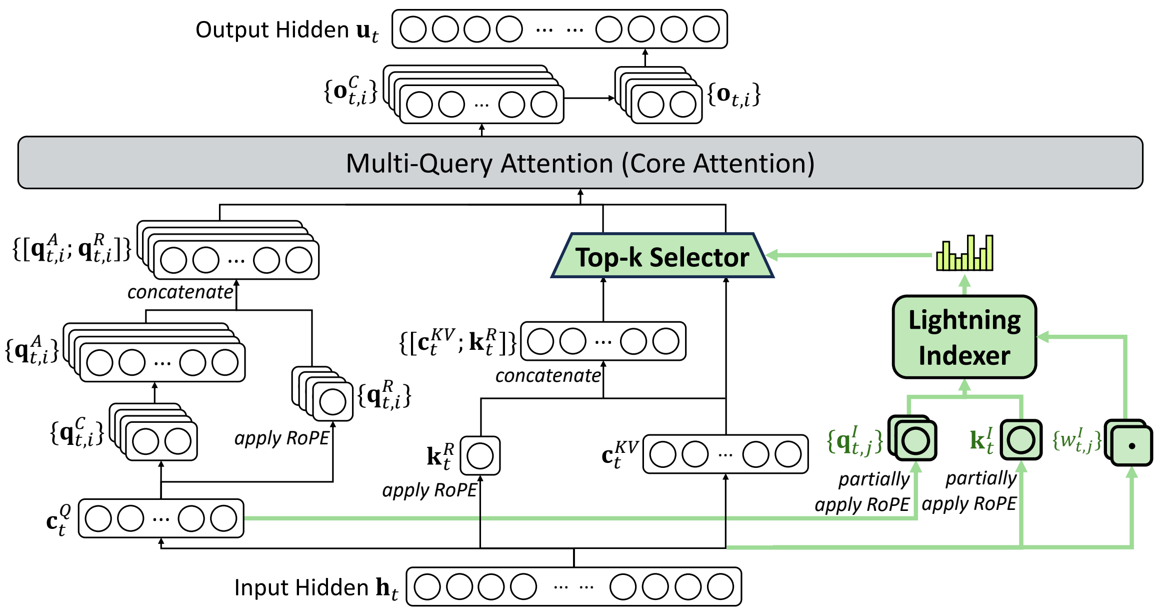


Figure 2 | Attention architecture of DeepSeek-V3.2, where DSA is instantiated under MLA. The green part illustrates how DSA selects the top-k key-value entries according to the indexer.

Dense Warm-up Stage. We first use a short warm-up stage to initialize the lightning indexer. In this stage, we keep dense attention and freeze all model parameters except for the lightning indexer. To align the indexer outputs with the main attention distribution, for the t-th query token, we first aggregate the main attention scores by summing across all attention heads. This sum is then L1-normalized along the sequence dimension to produce a target distribution $p_{t,:} \in \mathbb{R}^t$. Based on $p_{t,:}$, we set a KL-divergence loss as the training objective of the indexer:

$$\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,:} \,\|\, \mathrm{Softmax}(I_{t,:})\right). \tag{3}$$

For warm-up, we use a learning rate of 10^{-3}. We train the indexer for only 1000 steps, with each step consisting of 16 sequences of 128K tokens, resulting in a total of 2.1B tokens.

Sparse Training Stage. Following indexer warm-up, we introduce the fine-grained token selection mechanism and optimize all model parameters to adapt the model to the sparse pattern of DSA. In this stage, we also keep aligning the indexer outputs to the main attention distribution, but considering only the selected token set $\mathcal{S}_t = \{s \mid I_{t,s} \in \text{Top-k}(I_{t,:})\}$:

$$\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,\mathcal{S}_t} \,\|\, \mathrm{Softmax}(I_{t,\mathcal{S}_t})\right). \tag{4}$$

It is worth noting that we detach the indexer input from the computational graph for separate optimization. The training signal of the indexer is from only $\mathcal{L}^I$, while the optimization of the main model is according to only the language modeling loss. In this sparse training stage, we use a learning rate of 7.3×10^{-6}, and select 2048 key-value tokens for each query token. We train both the main model and the indexer for 15000 steps, with each step consisting of 480 sequences of 128K tokens, resulting in a total of 943.7B tokens.

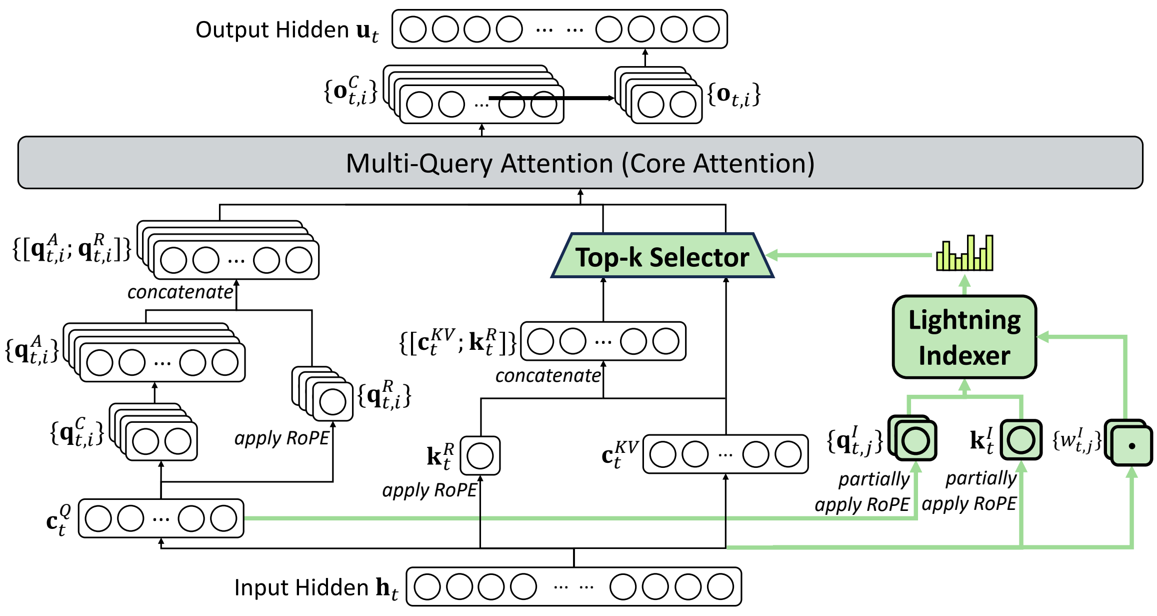


图2 DeepSeek-V3.2的注意力架构，其中DSA在MLA下实例化。绿色部分说明了 DSA 如何根据索引器选择前 k 个键值条目。

密集热身阶段。我们首先使用一个简短的预热阶段来初始化闪电索引器。在这个阶段，我们保持高度关注并冻结除闪电索引器之外的所有模型参数。为了将索引器输出与主要注意力分布对齐，对于第 t 个查询标记，我们首先通过对所有注意力头求和来聚合主要注意力分数。然后，该总和沿序列维度进行 L1 归一化，以生成目标分布 $p_{t,:} \in \mathbb{R}^t$。基于$p_{t,:}$，我们设置KL散度损失作为索引器的训练目标：

$$\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,:} \,\|\, \mathrm{Softmax}(I_{t,:})\right). \tag{3}$$

对于热身，我们使用 10^{-3} 的学习率。我们仅训练索引器 1000 个步骤，每个步骤由 16 个 128K 个令牌序列组成，总共产生 2.1B 个令牌。

稀疏训练阶段。在索引器预热之后，我们引入了细粒度标记选择机制并优化所有模型参数以使模型适应 DSA 的稀疏模式。在此阶段，我们还保持将索引器输出与主要注意力分布对齐，但仅考虑所选标记集 $\mathcal{S}_t = \{s \mid I_{t,s} \in \text{Top-k}(I_{t,:})\}$：

$$\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,\mathcal{S}_t} \,\|\, \mathrm{Softmax}(I_{t,\mathcal{S}_t})\right). \tag{4}$$

值得注意的是，我们将索引器输入与计算图分离以进行单独优化。索引器的训练信号仅来自$\mathcal{L}^I$，而主模型的优化仅根据语言建模损失。在这个稀疏训练阶段，我们使用 7.3×10^{-6} 的学习率，并为每个查询标记选择 2048 个键值标记。我们对主模型和索引器进行了 15000 个步骤的训练，每个步骤由 480 个 128K 个 token 的序列组成，总共得到 943.7B 个 token。

2.2. Parity Evaluation

Standard Benchmark In September 2025, we evaluate DeepSeek-V3.2-Exp on a suite of benchmarks, which focus on diverse capabilities, and compare it with DeepSeek-V3.1-Terminus showing similar performance. While DeepSeek V3.2 Exp significantly improves computational efficiency on long sequences, we do not observe substantial performance degradation compared with DeepSeek-V3.1-Terminus, on both short- and long-context tasks.

Human Preference Given that direct human preference assessments are inherently susceptible to bias, we employ ChatbotArena as an indirect evaluation framework to approximate user preferences for the newly developed base models. Both DeepSeek-V3.1-Terminus and DeepSeek-V3.2-Exp share an identical post-training strategy, and their Elo scores, obtained from evaluations conducted on 10 November 2025, are closely matched. These results suggest that the new base model achieves performance on par with the previous iteration, despite incorporating a sparse attention mechanism.

Long Context Eval Following the release of DeepSeek-V3.2-Exp, several independent long-context evaluations were conducted using previously unseen test sets. A representative benchmark is AA-LCR[3], in which DeepSeek-V3.2-Exp scores four points higher than DeepSeek-V3.1-Terminus in reasoning mode. In the Fiction.liveBench evaluation[4], DeepSeek-V3.2-Exp consistently outperforms DeepSeek-V3.1-Terminus across multiple metrics. This evidence indicates the base checkpoint of DeepSeek-V3.2-Exp does not regress on long context tasks.

2.3. Inference Costs

DSA reduces the core attention complexity of the main model from $O(L^2)$ to $O(Lk)$, where k ($\ll L$) is the number of selected tokens. Although the lightning indexer still has a complexity of $O(L^2)$, it requires much less computation compared with MLA in DeepSeek-V3.1-Terminus. Combined with our optimized implementation, DSA achieves a significant end-to-end speedup in long-context scenarios. Figure 3 presents how token costs of DeepSeek-V3.1-Terminus and DeepSeek-V3.2 vary with the token position in the sequence. These costs are estimated from benchmarking the actual service deployed on H800 GPUs, at a rental price of 2 USD per GPU hour. Note that for short-sequence prefilling, we specially implement a masked MHA mode to simulate DSA, which can achieve higher efficiency under short-context conditions.

3. Post-Training

After continued pre-training, we perform post-training to create the final DeepSeek-V3.2. The post-training of DeepSeek-V3.2 also employs sparse attention in the same way as the sparse continued pre-training stage. For DeepSeek-V3.2, we maintain the same post-training pipeline as in DeepSeek-V3.2-Exp, which includes specialist distillation and mixed RL training.

Specialist Distillation For each task, we initially develop a specialized model dedicated exclusively to that particular domain, with all specialist models being fine-tuned from the same

[3]https://artificialanalysis.ai/evaluations/artificial-analysis-long-context-reasoning

[4]https://fiction.live/stories/Fiction-liveBench-April-6-2025/oQdzQvKHw8JyXbN87

2.2.奇偶评估

标准基准 2025 年 9 月，我们在一套侧重于不同功能的基准上评估 DeepSeek-V3.2-Exp，并将其与显示相似性能的 DeepSeek-V3.1-Terminus 进行比较。虽然 DeepSeek V3.2 Exp 显着提高了长序列的计算效率，但在短上下文和长上下文任务上，与 DeepSeek-V3.1-Terminus 相比，我们没有观察到显着的性能下降。

人类偏好 鉴于直接的人类偏好评估本质上容易受到偏差的影响，我们采用 ChatbotArena 作为间接评估框架来近似新开发的基本模型的用户偏好。 DeepSeek-V3.1-Terminus 和 DeepSeek-V3.2-Exp 共享相同的训练后策略，并且从 2025 年 11 月 10 日进行的评估中获得的 Elo 分数非常匹配。这些结果表明，尽管采用了稀疏注意力机制，但新的基础模型的性能与之前的迭代相当。

长上下文评估 DeepSeek-V3.2-Exp 发布后，使用以前未见过的测试集进行了多次独立的长上下文评估。代表性的基准测试是 AA-LCR[3]，其中 DeepSeek-V3.2-Exp 在推理模式下的得分比 DeepSeek-V3.1-Terminus 高 4 分。在 Fiction.liveBench 评估[4]中，DeepSeek-V3.2-Exp 在多个指标上始终优于 DeepSeek-V3.1-Terminus。该证据表明 DeepSeek-V3.2-Exp 的基本检查点不会在长上下文任务上出现回归。

2.3.推理成本

DSA 将主模型的核心注意力复杂度从 $O(L^2)$ 降低到 $O(Lk)$，其中 k ($\ll L$) 是所选 token 的数量。尽管闪电索引器的复杂度仍然是$O(L^2)$，但与 DeepSeek-V3.1-Terminus 中的 MLA 相比，它需要的计算量要少得多。结合我们的优化实现，DSA 在长上下文场景中实现了显着的端到端加速。图 3 显示了 DeepSeek-V3.1-Terminus 和 DeepSeek-V3.2 的令牌成本如何随序列中令牌位置的变化而变化。这些成本是通过对 H800 GPU 上部署的实际服务进行基准测试估算得出的，租赁价格为每 GPU 小时 2 美元。请注意，对于短序列预填充，我们专门实现了一种 masked MHA 模式来模拟 DSA，这可以在短上下文条件下实现更高的效率。

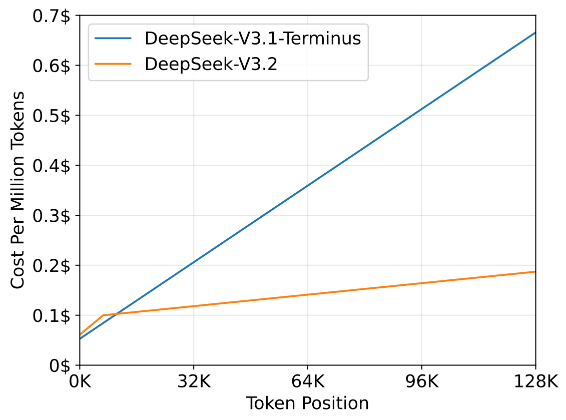
3. 训练后

经过持续的预训练后，我们进行后训练以创建最终的 DeepSeek-V3.2。 DeepSeek-V3.2的后训练也采用了稀疏注意力，与稀疏持续预训练阶段相同。对于 DeepSeek-V3.2，我们维护与 DeepSeek-V3.2-Exp 相同的训练后流程，其中包括专业蒸馏和混合 RL 训练。

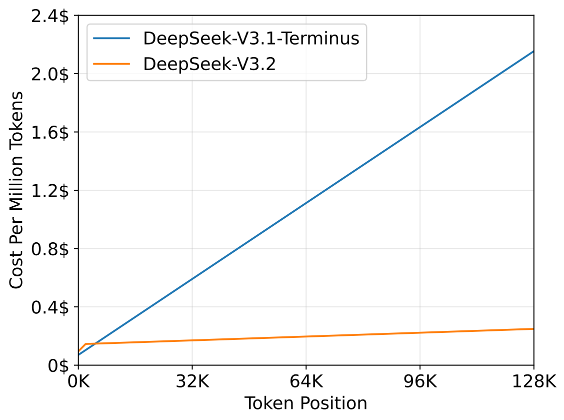
专业蒸馏对于每项任务，我们最初开发一个专门用于该特定领域的专业模型，所有专业模型都从相同的基础上进行微调

[3]https://artificialanalysis.ai/evaluations/artificial-analysis-long-context-reasoning

[4]https://fiction.live/stories/Fiction-liveBench-April-6-2025/oQdzQvKHw8JyXbN87



(a) Prefilling



(b) Decoding

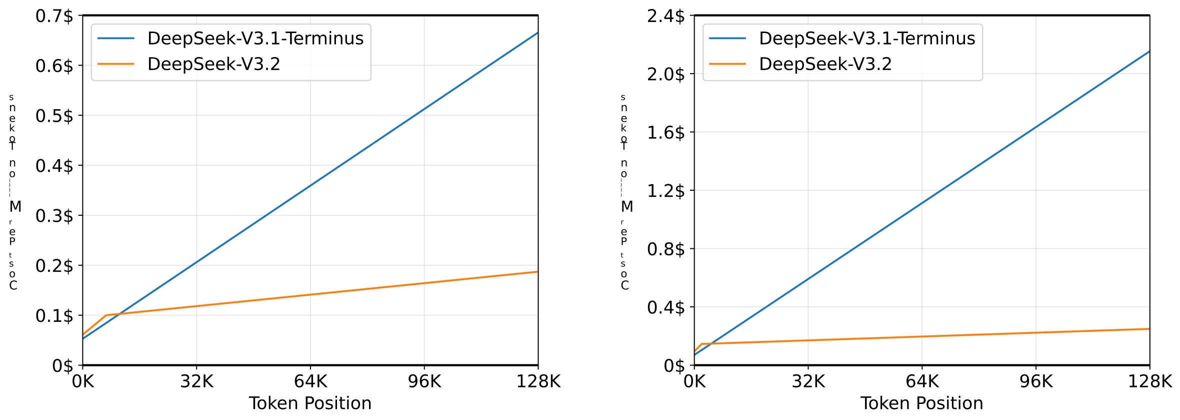
Figure 3 | Inference costs of DeepSeek-V3.1-Terminus and DeepSeek-V3.2 on H800 clusters.

pre-trained DeepSeek-V3.2 base checkpoint. In addition to writing tasks and general question-answering, our framework encompasses six specialized domains: mathematics, programming, general logical reasoning, general agentic tasks, agentic coding, and agentic search, with all the domains supporting both thinking and non-thinking modes. Each specialist is trained with large-scale Reinforcement Learning (RL) computing. Furthermore, we employ different models to generate training data for long chain-of-thought reasoning (thinking mode) and direct response generation (non-thinking mode). Once the specialist models are prepared, they are used to produce the domain-specific data for the final checkpoint. Experimental results demonstrate that models trained on the distilled data achieve performance levels only marginally below those of domain-specific specialists, with the performance gap being effectively eliminated through subsequent RL training.

Mixed RL Training For DeepSeek-V3.2, we still adopt Group Relative Policy Optimization (GRPO) (DeepSeek-AI, 2025; Shao et al., 2024) as the RL training algorithm. As DeepSeek-V3.2-Exp, we merge reasoning, agent, and human alignment training into one RL stage. This approach effectively balances performance across diverse domains while circumventing the catastrophic forgetting issues commonly associated with multi-stage training paradigms. For reasoning and agent tasks, we employ rule-based outcome reward, length penalty, and language consistency reward. For general tasks, we employ a generative reward model where each prompt has its own rubrics for evaluation.

DeepSeek-V3.2 and DeepSeek-V3.2-Speciale DeepSeek-V3.2 integrates reasoning, agent, and human alignment data distilled from specialists, undergoing thousands of steps of continued RL training to reach the final checkpoints. To investigate the potential of extended thinking, we also developed an experimental variant, DeepSeek-V3.2-Speciale. This model was trained exclusively on reasoning data with a reduced length penalty during RL. Additionally, we incorporated the dataset and reward method from DeepSeekMath-V2 (Shao et al., 2025) to enhance capabilities in mathematical proofs.

We would like to highlight our efforts in how to create a stable recipe to scale up RL compute in Section 3.1, and how to integrate thinking into agentic tasks in Section 3.2



(a) Prefilling (b) Decoding

图 3 | DeepSeek-V3.1-Terminus 和 DeepSeek-V3.2 在 H800 集群上的推理成本。

预训练的 DeepSeek-V3.2 基础检查点。除了编写任务和一般问答之外，我们的框架还涵盖六个专业领域：数学、编程、一般逻辑推理、一般代理任务、代理编码和代理搜索，所有领域都支持思维和非思维模式。每位专家都接受过大规模强化学习 (RL) 计算的培训。此外，我们采用不同的模型来生成长链思维推理（思维模式）和直接响应生成（非思维模式）的训练数据。一旦准备好专业模型，它们就会被用来为最终检查点生成特定于领域的数据。实验结果表明，在蒸馏数据上训练的模型所达到的性能水平仅略低于特定领域专家的性能水平，并且通过后续的 RL 训练有效地消除了性能差距。

混合强化学习训练对于 DeepSeek-V3.2，我们仍然采用组相对策略优化（GRPO）（DeepSeek-AI，2025；Shao et al.，2024）作为强化学习训练算法。作为 DeepSeek-V3.2-Exp，我们将推理、代理和人类对齐训练合并到一个 RL 阶段。这种方法有效地平衡了不同领域的性能，同时避免了通常与多阶段训练范例相关的灾难性遗忘问题。对于推理和代理任务，我们采用基于规则的结果奖励、长度惩罚和语言一致性奖励。对于一般任务，我们采用生成奖励模型，其中每个提示都有自己的评估规则。

DeepSeek-V3.2 和 DeepSeek-V3.2-Speciale DeepSeek-V3.2 集成了专家提取的推理、代理和人类对齐数据，经过数千步的持续 RL 训练以达到最终检查点。为了研究扩展思维的潜力，我们还开发了一个实验变体，DeepSeek-V3.2-Speciale。该模型专门针对推理数据进行训练，并在强化学习期间减少了长度损失。此外，我们还结合了 DeepSeekMath-V2 (Shao et al., 2025) 的数据集和奖励方法来增强数学证明的能力。

我们想在第 3.1 节中强调我们在如何创建稳定的方法来扩展 RL 计算方面所做的努力，以及在第 3.2 节中如何将思维整合到代理任务中

3.1. Scaling GRPO

We first review the objective of GRPO. GRPO optimizes the policy model π_θ by maximizing the following objective on a group of responses $\{o_1, \cdots, o_G\}$ sampled from the old policy π_{old} given each question q:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^G\sim\pi_{\text{old}}(\cdot|q)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}$$
$$\min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t})\,\|\,\pi_{\text{ref}}(o_{i,t})\right)\Bigg], \tag{5}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\text{old}}(o_{i,t}|q, o_{i,<t})} \tag{6}$$

is the importance sampling ratio between the current and old policy. ε and β are hyper-parameters controlling the clipping range and KL penalty strength, respectively. $\hat{A}_{i,t}$ is the advantage of $o_{i,t}$ which is estimated by normalizing the outcome reward within each group. Specifically, a set of reward models are used to score an outcome reward R_i for each output o_i in the group, yielding G rewards $\boldsymbol{R} = \{R_1, \cdots, R_G\}$ respectively. The advantage of $o_{i,t}$ is calculated by subtracting the average reward of the group from the reward of output o_i, i.e., $\hat{A}_{i,t} = R_i - \text{mean}(\boldsymbol{R})$.

In the following, we outline additional strategies that stabilize RL scaling, directly building on the GRPO algorithm.

Unbiased KL Estimate Given $o_{i,t}$ is sampled from the old policy $\pi_{\text{old}}(\cdot|q, o_{i,<t})$, we correct the K3 estimator (Schulman, 2020) to obtain an unbiased KL estimate using the importance-sampling ratio between the current policy π_θ and the old policy π_{old} .

$$\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t})\,\|\,\pi_{\text{ref}}(o_{i,t})\right) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\text{old}}(o_{i,t}|q, o_{i,<t})}\left(\frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - \log\frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - 1\right). \tag{7}$$

As a direct result of this adjustment, the gradient of this KL estimator becomes unbiased, which eliminates systematic estimation errors, thereby facilitating stable convergence. This contrasts sharply with the original K3 estimator, particularly when the sampled tokens have substantially lower probabilities under the current policy than the reference policy, i.e., $\pi_\theta \ll \pi_{\text{ref}}$. In such cases, the gradient of the K3 estimator assigns disproportionately large, unbounded weights to maximize the likelihood of these tokens, resulting in noisy gradient updates that accumulate to degrade sample quality in subsequent iterations and lead to unstable training dynamics. In practice, we find that different domains benefit from varying strengths of KL regularization. For certain domains, such as mathematics, applying a relatively weak KL penalty or even omitting it entirely can yield improved performance.

Off-Policy Sequence Masking To improve the efficiency of RL systems, we typically generate a large batch of rollout data, which is subsequently split into multiple mini-batches for several gradient update steps. This practice inherently introduces off-policy behavior. Additionally, inference frameworks used for efficient data generation are often highly optimized, which may differ in implementation details from training frameworks. Such training-inference inconsistency

3.1.扩展GRPO

我们首先回顾GRPO 的目标。 GRPO 通过在给定每个问题 q 的旧策略 π_{old} 中采样的一组响应 $\{o_1$、$\cdots$ 和 $o_G\}$ 上最大化以下目标来优化策略模型 π_θ：

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|q)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}$$

$$\min\left(r_{i,t}(\theta)\hat{A}_{i,t},\text{clip}\left(r_{i,t}(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_{i,t}\right)-\beta\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t})\,\|\,\pi_{\text{ref}}(o_{i,t})\right)\Bigg], \qquad (5)$$

在哪里

$$r_{i,t}(\theta)=\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\text{old}}(o_{i,t}|q,o_{i,<t})} \qquad (6)$$

是当前策略与旧策略之间的重要性采样比。 ε和β是分别控制裁剪范围和KL惩罚强度的超参数。 $\hat{A}_{i,t}$ 是 $o_{i,t}$ 的优势，它是通过标准化每组内的结果奖励来估计的。具体来说，一组奖励模型用于为组中的每个输出 o_i 评分结果奖励 R_i，分别产生 G 奖励 $\boldsymbol{R}=\{R_1$、$\cdots$、$R_G\}$。 $o_{i,t}$的优势是通过输出o_i的奖励减去该组的平均奖励来计算的，即$\hat{A}_{i,t}=R_i-$平均值($\boldsymbol{R}$)。

接下来，我们将概述直接基于 GRPO 算法稳定 RL 缩放的其他策略。

无偏 KL 估计 给定 $o_{i,t}$ 是从旧策略 $\pi_{\text{old}}(\cdot|q$、$o_{i,<t})$ 中采样的，我们校正 K3 估计器（Schulman，2020），以使用当前策略 π_θ 和旧策略 π_{old} 之间的重要性采样比率来获得无偏 KL 估计。

$$\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t})\,\|\,\pi_{\text{ref}}(o_{i,t})\right)=\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\text{old}}(o_{i,t}|q,o_{i,<t})}\left(\frac{\pi_{\text{ref}}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})}-\log\frac{\pi_{\text{ref}}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})}-1\right). \qquad (7)$$

作为这种调整的直接结果，该 KL 估计器的梯度变得无偏，从而消除了系统估计误差，从而促进稳定收敛。这与原始 K3 估计器形成鲜明对比，特别是当采样令牌在当前策略下的概率远低于参考策略（即 $\pi_\theta \ll \pi_{\text{ref}}$）时。在这种情况下，K3 估计器的梯度会分配不成比例的大的无界权重，以最大化这些标记的可能性，从而导致噪声梯度更新累积，从而降低后续迭代中的样本质量，并导致不稳定的训练动态。在实践中，我们发现不同的领域受益于不同的 KL 正则化强度。对于某些领域，例如数学，应用相对较弱的 KL 惩罚甚至完全忽略它可以提高性能。

离策略序列屏蔽 为了提高 RL 系统的效率，我们通常会生成大量的 rollout 数据，随后将其分为多个小批量，以进行多个梯度更新步骤。这种做法本质上引入了离策略行为。此外，用于高效数据生成的推理框架通常是高度优化的，其实现细节可能与训练框架不同。这种训练-推理不一致的情况

further exacerbates the degree of off-policyness. To stabilize training and improve tolerance for off-policy updates, we mask negative sequences that introduce significant policy divergence, as measured by the KL divergence between the data-sampling policy π_{old} and the current policy π_θ. More specifically, we introduce a binary mask M into the GRPO loss:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\text{old}}(\cdot|q)} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) M_{i,t} - \beta \mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t}) \,\|\, \pi_{\text{ref}}(o_{i,t})\right) \Bigg], \tag{8}$$

where

$$M_{i,t} = \begin{cases} 0 & \hat{A}_{i,t} < 0, \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \log \frac{\pi_{\text{old}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} > \delta \\ 1 & \text{otherwise,} \end{cases} \tag{9}$$

and δ is a hyper-parameter that controls the threshold of policy divergence. Note that π_{old} here denotes the sampling probability directly returned by the inference framework, thus the KL divergence between the old and current policy accounts for both sources of off-policyness mentioned above. It is also worth noting that we only mask sequences with negative advantages. Intuitively, the model benefits the most by learning from its own mistakes, whereas highly off-policy negative samples can be detrimental, potentially misleading or destabilizing the optimization process. We empirically observe that this Off-Policy Sequence Masking operation improves stability in certain training scenarios that would otherwise exhibit instability.

Keep Routing Mixture-of-Experts (MoE) models improve computational efficiency by activating only a subset of expert modules during inference. However, discrepancies between inference and training frameworks, compounded by policy updates, can result in inconsistent expert routing during inference and training even for identical inputs. Such inconsistency induces abrupt shifts in the active parameter subspace, which destabilizes optimization and exacerbates off-policy issues. To mitigate this, we preserve the expert routing paths used during sampling in the inference framework and enforce the same routing paths during training, ensuring that identical expert parameters are optimized. This Keep Routing operation was found crucial for RL training stability of MoE models, and has been adopted in our RL training pipeline since DeepSeek-V3-0324.

Keep Sampling Mask Top-p and top-k sampling are widely used sampling strategies to enhance the quality of responses generated by LLMs. Employing these strategies in RL training is also advantageous, as it avoids sampling extremely low-probability tokens that would be used as optimization targets. While such truncation preserves sample quality, it introduces a mismatch between the action spaces of π_{old} and π_θ, which violates the principles of importance sampling and destabilizes training. To address this, we preserve the truncation masks during sampling from π_{old} and apply them to π_θ during training, ensuring both policies share identical action subspaces. Empirically, we find that combining top-p sampling with the Keep Sampling Mask strategy effectively preserves language consistency during RL training.

进一步加剧了离政策的程度。为了稳定训练并提高对离策略更新的容忍度，我们屏蔽了引入显着策略分歧的负序列，如数据采样策略 π_{old} 和当前策略 π_θ 之间的 KL 分歧来衡量。更具体地说，我们在 GRPO 损失中引入二进制掩码 M:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|q)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right)M_{i,t} - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t})\,\middle\|\,\pi_{\text{ref}}(o_{i,t})\right)\Bigg], \quad (8)$$

在哪里

$$M_{i,t} = \begin{cases} 0 & \hat{A}_{i,t} < 0, \frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\log\frac{\pi_{\text{old}}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} > \delta \\ 1 & \text{otherwise}, \end{cases} \quad (9)$$

δ是控制策略发散阈值的超参数。请注意，这里的 π_{old} 表示推理框架直接返回的采样概率，因此旧策略和当前策略之间的 KL 差异解释了上述两种离策略的来源。还值得注意的是，我们只屏蔽具有负面优势的序列。直观上，模型通过从自己的错误中学习而获益最多，而高度偏离策略的负样本可能是有害的，可能会误导或破坏优化过程的稳定性。我们凭经验观察到，这种离策略序列屏蔽操作提高了某些训练场景中的稳定性，否则这些场景会表现出不稳定。

Keep Routing 专家混合 (MoE) 模型通过在推理过程中仅激活专家模块的子集来提高计算效率。然而，推理和训练框架之间的差异，加上策略更新，可能会导致推理和训练期间专家路由不一致，即使对于相同的输入也是如此。这种不一致会导致活动参数子空间突然发生变化，从而破坏优化的稳定性并加剧离策略问题。为了缓解这种情况，我们在推理框架中保留采样期间使用的专家路由路径，并在训练期间强制执行相同的路由路径，确保优化相同的专家参数。人们发现，这种“保持路由”操作对于 MoE 模型的 RL 训练稳定性至关重要，并且自 DeepSeek-V3-0324 以来一直在我们的 RL 训练流程中采用。

保持采样掩码 Top-p 和 top-k 采样是广泛使用的采样策略，用于提高 LLM 生成的响应的质量。在 RL 训练中采用这些策略也是有利的，因为它可以避免采样将用作优化目标的极低概率标记。虽然这种截断保留了样本质量，但它在 π_{old} 和 π_θ 的动作空间之间引入了不匹配，这违反了重要性采样的原则并破坏了训练的稳定性。为了解决这个问题，我们在从 π_{old} 采样期间保留截断掩码，并在训练期间将它们应用于 π_θ，确保两个策略共享相同的动作子空间。根据经验，我们发现将 top-p 采样与保持采样掩模策略相结合可以有效地保持 RL 训练期间的语言一致性。

3.2. Thinking in Tool-Use

3.2.1. Thinking Context Management

DeepSeek-R1 has demonstrated that incorporating a thinking process can significantly enhance a model's ability to solve complex problems. Building on this insight, we aim to integrate thinking capabilities into tool-calling scenarios.

We observed that replicating DeepSeek-R1's strategy—discarding reasoning content upon the arrival of the second round of messages—results in significant token inefficiency. This approach forces the model to redundantly re-reason through the entire problem for each subsequent tool call. To mitigate this, we developed a context management strictly tailored for tool-calling scenarios as shown in Fig 4:

- Historical reasoning content is discarded only when a new **user message** is introduced to the conversation. If only tool-related messages (e.g., tool outputs) are appended, the reasoning content is **retained** throughout the interaction.
- When reasoning traces are removed, the history of **tool calls and their results** remains preserved in the context.

Notably, certain agent frameworks, such as Roo Code or Terminus, simulate tool interactions via user messages. These frameworks may not fully benefit from our enhanced reasoning persistence due to the context management rules outlined above. Therefore, we recommend utilizing non-thinking models for optimal performance with such architectures.

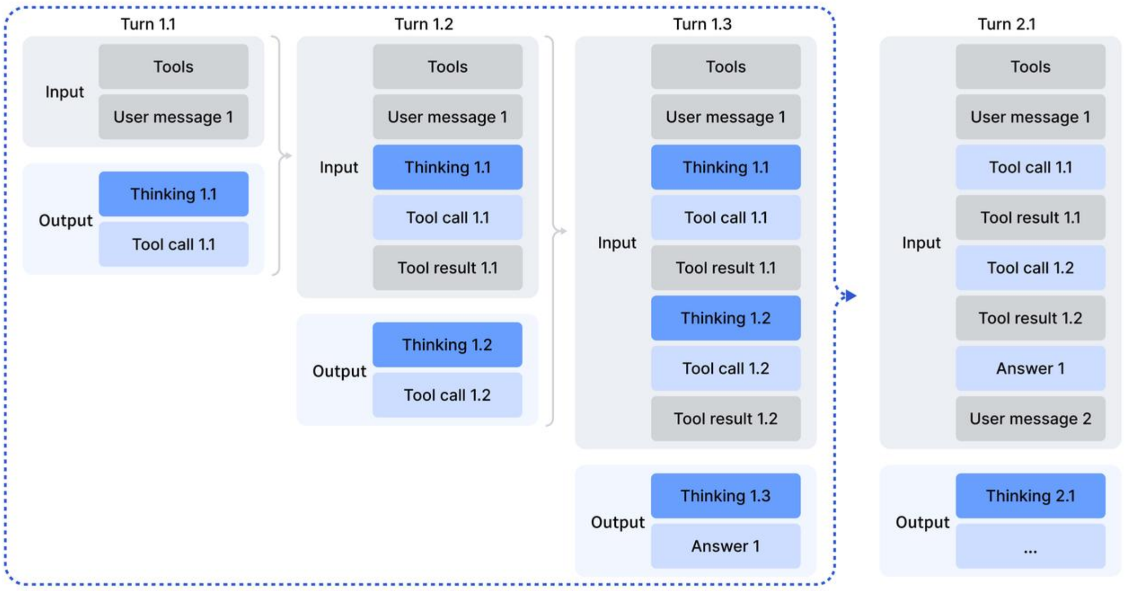


Figure 4 | Thinking retention mechanism in tool-calling scenarios.

3.2.2. Cold-Start

Given the availability of reasoning data (non-agentic) and non-reasoning agentic data, a straightforward strategy for integrating these two capabilities is through carefully designed prompting. We posit that the model possesses sufficient ability to accurately follow explicit instructions, thereby enabling the seamless incorporation of tool execution within the reasoning process.

3.2.工具使用的思考

3.2.1. Thinking Context Management

DeepSeek-R1 已经证明，整合思维过程可以显着增强模型解决复杂问题的能力。基于这种洞察，我们的目标是将思维能力集成到工具调用场景中。

我们观察到，复制 DeepSeek-R1 的策略（在第二轮消息到达时丢弃推理内容）会导致明显的令牌效率低下。这种方法迫使模型为每个后续工具调用对整个问题进行冗余地重新推理。为了缓解这个问题，我们开发了专门针对工具调用场景定制的上下文管理，如图 4 所示：

- 仅当新的用户消息引入对话时，历史推理内容才会被丢弃。如果仅附加与工具相关的消息（例如，工具输出），则在整个交互过程中保留推理内容。
- 当推理痕迹被删除时，工具调用的历史记录及其结果仍保留在上下文中。

值得注意的是，某些代理框架（例如 Roo Code 或 Terminus）通过用户消息模拟工具交互。由于上述上下文管理规则，这些框架可能无法完全受益于我们增强的推理持久性。因此，我们建议利用非思维模型来实现此类架构的最佳性能。

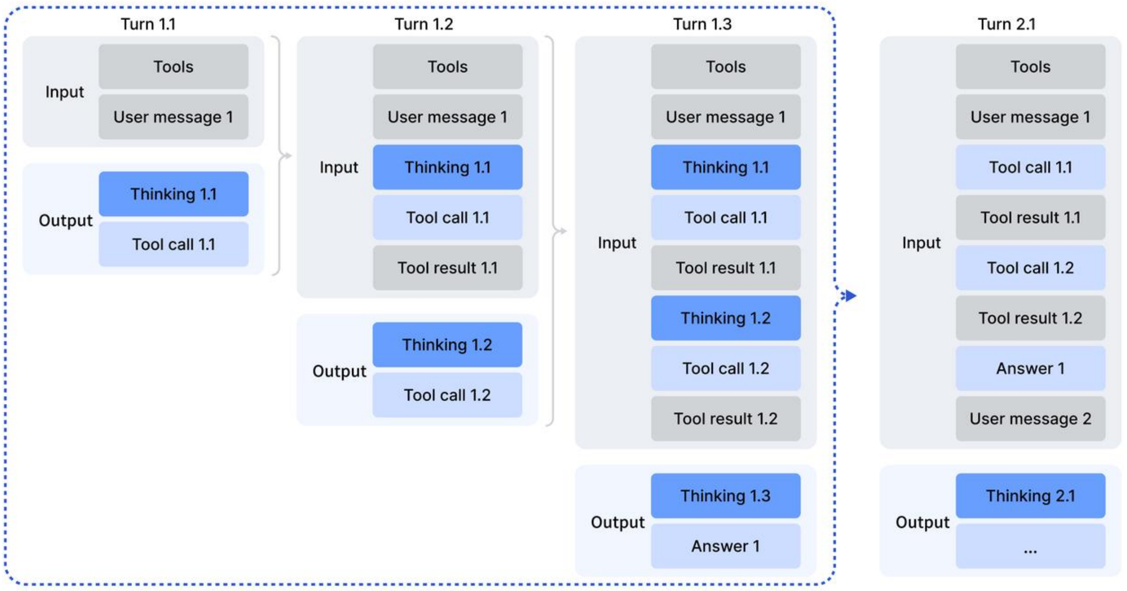


图4 | 工具调用场景中的思维保留机制 纳里奥斯。

3.2.2. Cold-Start

考虑到推理数据（非代理）和非推理代理数据的可用性，集成这两种功能的直接策略是通过精心设计的提示。我们认为该模型具有足够的能力来准确遵循明确的指令，从而能够将工具执行无缝地融入推理过程中。

To demonstrate the operation of the cold-start mechanism, we selectively sample the training data as shown in Appendix Tables 6–8. It is important to note that distinct task prompts are associated with different system prompts. Tables 6–8 present an illustrative example corresponding to a competitive programming prompt. Table 6 presents an example of our reasoning data, which uses a system prompt to explicitly asks the model to do reasoning before the final answer and uses a special tag <think></think> to label the reasoning path. Table 7 shows the prompt of non-reasoning agentic data, where the system prompt contains the guidance of toolcall. Table 8 presents the system prompt we designed to instruct the model to incorporate multiple tool calls within its reasoning process.

In this manner, although the reasoning in tool-use patterns may lack robustness, the model is occasionally able to generate the desired trajectories, thereby providing a basis for subsequent reinforcement learning stages.

3.2.3. Large-Scale Agentic Tasks

A diverse set of RL tasks is crucial for enhancing model robustness. For tasks such as search, code engineering, and code interpretation, we employ real-world tools, including actual web search APIs, coding tools, and Jupyter Notebooks. While these RL environments are real, the prompts employed are either extracted from Internet sources or synthetically generated, rather than obtained from actual user interactions. For other tasks, the environment and prompts are both synthetically constructed. The agent tasks we used are described in Table 1.

Table 1 | The description of different agent tasks, including the number of tasks, environment type (real or synthesized), and prompt source (extracted or synthesized).

	number of tasks	environment	prompt
code agent	24667	real	extracted
search agent	50275	real	synthesized
general agent	4417	synthesized	synthesized
code interpreter	5908	real	extracted

Search Agent We employ a multi-agent pipeline based on DeepSeek-V3.2 to generate diverse, high-quality training data. We first sample informative long-tail entities across diverse domains from large-scale web corpora. A question-construction agent then explores each entity using search tools with configurable depth and breadth parameters, consolidating the discovered information into question-answer pairs. Multiple answer-generation agents with heterogeneous configurations (different checkpoints, system prompts, etc.) produce diverse candidate responses for each proposed QA pair. A verification agent with search capabilities validates all answers through multiple passes, retaining only samples where the ground-truth is correct and all candidates are verifiably incorrect. These data spans multiple languages, domains, and difficulty levels. To complement these verifiable samples and better reflect real-world usage, we also augment the dataset with filtered instances from our existing helpful RL datasets, for which the search tool provides measurable benefits. We then develop detailed evaluation rubrics across multiple quality dimensions and employ a generative reward model to score responses based on these rubrics. This hybrid approach enables optimization for both factual reliability and practical helpfulness.

为了演示冷启动机制的操作，我们选择性地对训练数据进行采样，如附录表 6-8 所示。需要注意的是，不同的任务提示与不同的系统提示相关联。表 6-8 提供了与竞争性编程提示相对应的说明性示例。表 6 展示了我们的推理数据的示例，它使用系统提示明确要求模型在最终答案之前进行推理，并使用特殊标签 <think></think> 来标记推理路径。表7为非推理代理数据的提示，其中系统提示包含工具调用的指导。表 8 展示了我们设计的系统提示，用于指示模型在其推理过程中合并多个工具调用。

这样，虽然工具使用模式的推理可能缺乏鲁棒性，但模型偶尔能够生成所需的轨迹，从而为后续的强化学习阶段提供基础。

3.2.3. Large-Scale Agentic Tasks

多样化的强化学习任务对于增强模型的鲁棒性至关重要。对于搜索、代码工程和代码解释等任务，我们使用现实世界的工具，包括实际的网络搜索 API、编码工具和 Jupyter Notebook。虽然这些 RL 环境是真实的，但所使用的提示要么是从互联网资源中提取的，要么是综合生成的，而不是从实际的用户交互中获得的。对于其他任务，环境和提示都是综合构建的。我们使用的代理任务如表 1 所示。

表1 | 不同代理任务的描述，包括任务数量、环境类型（真实或合成）和提示源（提取或合成）。

	number of tasks	environment	prompt
code agent	24667	real	extracted
search agent	50275	real	synthesized
general agent	4417	synthesized	synthesized
code interpreter	5908	real	extracted

搜索代理我们采用基于 DeepSeek-V3.2 的多代理管道来生成多样化的高质量训练数据。我们首先从大型网络语料库中跨不同领域的信息长尾实体进行采样。然后，问题构建代理使用具有可配置深度和广度参数的搜索工具探索每个实体，将发现的信息合并到问题-答案对中。具有异构配置（不同的检查点、系统提示等）的多个答案生成代理为每个提出的 QA 对产生不同的候选响应。具有搜索功能的验证代理通过多次传递验证所有答案，仅保留基本事实正确且所有候选都可验证错误的样本。这些数据跨越多种语言、领域和难度级别。为了补充这些可验证的样本并更好地反映现实世界的使用情况，我们还使用现有有用的强化学习数据集中的过滤实例来扩充数据集，为此搜索工具提供了可衡量的好处。然后，我们制定跨多个质量维度的详细评估标准，并采用生成奖励模型根据这些标准对响应进行评分。这种混合方法可以优化事实可靠性和实际帮助。

Code Agent We constructed large-scale, executable environments for software issue resolution by mining millions of issue-Pull Request (PR) pairs from GitHub. This dataset was rigorously filtered using heuristic rules and LLM-based judgments to ensure high quality, requiring that each entry contain a reasonable issue description, a correlated gold patch, and a test patch for validation. An automated environment-setup agent, powered by DeepSeek-V3.2, was employed to build executable environments for these pairs. This agent handles package installation, dependency resolution, and test execution. Test results are output in the standard JUnit format, ensuring consistent parsing across programming languages and test frameworks. An environment is deemed successfully built only when applying the gold patch results in a non-zero count of false-to-positive (F2P) test cases (indicating the issue is fixed) and a zero count of pass-to-fail (P2F) test cases (indicating no regressions). Using this pipeline, we successfully built tens of thousands of reproducible issue resolution environments spanning multiple programming languages, including Python, Java, JavaScript, TypeScript, C, C++, Go, and PHP.

Code Interpreter Agent We utilize Jupyter Notebook as a code interpreter to address complex reasoning tasks. To facilitate this, we curate a diverse set of problems spanning mathematics, logic, and data science, each requiring the model to leverage code execution capabilities to arrive at a solution.

General Agent To scale up agent environments and tasks in RL, we employ an automatic environment-synthesis agent that synthesizes 1,827 task-oriented environments. These tasks are hard to solve but easy to verify. The synthesis workflow primarily consists of environment and toolset construction, task synthesis, and solution generation. Specifically, the workflow proceeds as follows.

1. Given a task category (e.g., planning a travel itinerary) and a sandbox equipped with a bash and a search tool, the agent first uses these tools to generate or retrieve relevant data from the Internet and store them in the sandbox database.
2. The agent then synthesizes a set of task-specific tools, each implemented as a function.
3. To create tasks that are both challenging and automatically verifiable, the agent initially proposes a simple task based on the current database, along with its solution and verification functions implemented in Python. The solution function is restricted to invoking tool functions or performing logical computations, and cannot call other functions or directly access the database, ensuring the task can only be solved through the tool interface. Additionally, the results produced by the solution function must be validated by the verification function. If the solution is not validated, the agent will modify the solution or verification functions until the solution's output passes the verification. The agent then iteratively increases the difficulty of the task and updates the corresponding solution and verification functions. During this iterative process, if the current toolset is not sufficient to solve the task, the agent will augment the toolset.

Following this workflow, we obtain thousands of <environment, tools, task, verifier> tuples. We then perform RL on this dataset using DeepSeek-V3.2 and retain only instances with non-zero pass@100, resulting in 1,827 environments and their corresponding tasks (4,417 in total). A synthetic trip-planning example is illustrated below. This example highlights that, while searching the large combinatorial space for a trip plan that satisfies all constraints is challenging, checking whether a given candidate solution satisfies these constraints is relatively straightforward.

代码代理 我们通过从 GitHub 挖掘数百万个问题拉取请求 (PR) 对，构建了用于软件问题解决的大规模可执行环境。该数据集使用启发式规则和基于 LLM 的判断进行严格过滤，以确保高质量，要求每个条目包含合理的问题描述、相关的黄金补丁和用于验证的测试补丁。采用由 DeepSeek-V3.2 提供支持的自动环境设置代理来为这些对构建可执行环境。该代理处理包安装、依赖关系解析和测试执行。测试结果以标准 JUnit 格式输出，确保跨编程语言和测试框架的一致解析。仅当应用黄金补丁导致假阳性 (F2P) 测试用例的非零计数（表明问题已修复）和通过到失败 (P2F) 测试用例的零计数（表明没有回归）时，才认为环境已成功构建。使用此管道，我们成功构建了数万个可重现的问题解决环境，涵盖多种编程语言，包括 Python、Java、JavaScript、TypeScript、C、C++、Go 和 PHP。

代码解释器代理我们利用 Jupyter Notebook 作为代码解释器来解决复杂的推理任务。为了实现这一目标，我们策划了一系列涵盖数学、逻辑和数据科学的问题，每个问题都需要模型利用代码执行功能来得出解决方案。

通用代理 为了扩展 RL 中的代理环境和任务，我们采用了自动环境合成代理，它可以合成 1,827 个面向任务的环境。这些任务很难解决，但很容易验证。综合工作流程主要包括环境和工具集构建、任务综合和解决方案生成。具体来说，工作流程如下进行。

1. 给定一个任务类别（例如，规划旅行行程）和一个配备 bash 和搜索工具的沙箱，代理首先使用这些工具从互联网生成或检索相关数据并将其存储在沙箱数据库中。 2. 然后，代理综合一组特定于任务的工具，每个工具都作为一个函数实现。 3. 为了创建既具有挑战性又可自动验证的任务，代理首先根据当前数据库提出一个简单的任务，以及用Python实现的解决方案和验证功能。求解功能仅限于调用工具函数或进行逻辑计算，不能调用其他函数或直接访问数据库，保证任务只能通过工具接口解决。此外，求解函数产生的结果必须由验证函数进行验证。如果解决方案未经过验证，代理将修改解决方案或验证函数，直到解决方案的输出通过验证。然后代理迭代地增加任务的难度并更新相应的解决方案和验证函数。在此迭代过程中，如果当前工具集不足以解决任务，代理将增强工具集。

按照此工作流程，我们获得了数千个<环境、工具、任务、验证者>元组。然后，我们使用 DeepSeek-V3.2 对此数据集执行强化学习，并仅保留非零 pass@100 的实例，从而产生 1,827 个环境及其相应的任务（总共 4,417 个）。下面说明了一个综合旅行计划的示例。这个例子强调，虽然在大型组合空间中搜索满足所有约束的旅行计划具有挑战性，但检查给定的候选解决方案是否满足这些约束相对简单。

An Example of Synthesized Task: Trip Planning

I'm planning a three-day trip starting from Hangzhou, and I need help creating an itinerary from October 1st to October 3rd, 2025. A few important requirements: I don't want to repeat any cities, hotels, attractions, or restaurants during the entire trip. Also, please make sure that every hotel, restaurant, and attraction you recommend is actually located in the city where I'll be staying that day. One more thing about the second day - I'm trying to be smart about my budget. If I end up booking a luxury hotel that costs 800 CNY or more per night, then I need to be more careful with other expenses: my total spending on both restaurants (lunch and dinner) should stay under 350 CNY, both restaurants should be rated at least 4.0 stars, and the afternoon attraction ticket needs to be less than 120 CNY. If the hotel on day 2 is in the mid-to-high range (500-800 CNY), then I have a bit more flexibility - I just need to make sure at least one of my restaurant choices is rated 4.0 or higher, and the attraction ticket should be below 180 CNY. For more affordable hotels (200-500 CNY range), I only need to ensure that at least one restaurant has a rating of 3.2 or above. Can you help me put together this itinerary?

Submit Result Format

[
{ "time": "2025-10-01", "city": "cite_name", "hotel": "hotel_name", "afternoon_restaurant": "restaurant_name", "afternoon_attraction": "attraction_name", "evening_restaurant": "restaurant_name" },
{ "time": "2025-10-02", "city": "cite_name", "hotel": "hotel_name", "afternoon_restaurant": "restaurant_name", "afternoon_attraction": "attraction_name", "evening_restaurant": "restaurant_name" },
{ "time": "2025-10-03", "city": "cite_name", "hotel": "hotel_name", "afternoon_restaurant": "restaurant_name", "afternoon_attraction": "attraction_name", "evening_restaurant": "restaurant_name" }
]

Tool Set for Trip Planning

Function Name	Description
`get_all_attractions_by_city(city)`	Get all attractions for given city.
`get_all_cities()`	Get all cities from the database.
`get_all_hotels_by_city(city)`	Get all hotels for given city.
`get_all_restaurants_by_city(city)`	Get all restaurants for given city.
`get_city_by_attraction(attraction)`	Get city for given attraction name.
`get_city_by_hotel(hotel)`	Get city for given hotel name.
`get_city_by_restaurant(restaurant)`	Get city for given restaurant name.
`get_city_transport(city)`	Get all intra-city transport options for given city.
`get_infos_by_attraction(info_keywords, attraction)`	Get specified infos for given attraction.
`get_infos_by_city(info_keywords, city)`	Get specified infos for given city.
`get_infos_by_hotel(info_keywords, hotel)`	Get specified infos for given hotel.
`get_infos_by_restaurant(info_keywords, restaurant)`	Get specified infos for given restaurant.
`get_inter_city_transport(from_city, to_city)`	Get all transports between given city pair.
`get_weather_by_city_date(city, date)`	Get weather for given city-date pair.
`submit_result(answer_text)`	Submit the final answer content.

4. Evaluation

4.1. Main Results

We evaluate models on MMLU-Pro (Wang et al., 2024), GPQA Diamond (Rein et al., 2023), Human Last Exam (HLE) Text-only (Phan et al., 2025), LiveCodeBench (2024.08-2025.04), Code-

综合任务的示例：旅行计划

我计划从杭州出发为期三天的旅行，需要帮助制定2025年10月1日至10月3日的行程。有几个重要的要求：我不想在整个行程中重复任何城市、酒店、景点或餐厅。另外，请确保您推荐的每家酒店、餐厅和景点实际上都位于我当天要入住的城市。关于第二天的另一件事 - 我正在努力明智地对待我的预算。如果我最终预订的是每晚800元或以上的豪华酒店，那么我需要更加小心其他费用：我在两家餐厅（午餐和晚餐）的总花费应控制在350元以内，两家餐厅的评级至少应为4.0星，下午景点门票需低于120元。如果第二天的酒店是中高档酒店（500-800元），那么我就有更大的灵活性——我只需要确保我选择的至少一家餐厅评级为4.0或更高，并且景点门票应该低于180元。对于更实惠的酒店（200-500元人民币范围），我只需要确保至少有一家餐厅的评级为3.2或以上。你能帮我安排一下这个行程吗?

提交结果格式

[
{ "time": "2025-10-01", "city": "cite_name", "hotel": "hotel_name", "afternoon_restaurant": "restaurant_name", "afternoon_attraction": "attraction_name", "evening_restaurant": "restaurant_name" }, { "time": "2025-10-02", "city": "cite_name", "hotel": "hotel_name", "afternoon_restaurant": "restaurant_name", "afternoon_attraction": "attraction_name", "evening_restaurant": "restaurant_name" }, { "time": "2025-10-03", "city": "cite_name", "hotel": "hotel_name", "afternoon_restaurant": "restaurant_name", "afternoon_attraction": "attraction_name", "evening_restaurant": "restaurant_name" }]

旅行计划工具集

Function Name	**Description**
`get_all_attractions_by_city(city)`	Get all attractions for given city.
`get_all_cities()`	Get all cities from the database.
`get_all_hotels_by_city(city)`	Get all hotels for given city.
`get_all_restaurants_by_city(city)`	Get all restaurants for given city.
`get_city_by_attraction(attraction)`	Get city for given attraction name.
`get_city_by_hotel(hotel)`	Get city for given hotel name.
`get_city_by_restaurant(restaurant)`	Get city for given restaurant name.
`get_city_transport(city)`	Get all intra-city transport options for given city.
`get_infos_by_attraction(info_keywords, attraction)`	Get specified infos for given attraction.
`get_infos_by_city(info_keywords, city)`	Get specified infos for given city.
`get_infos_by_hotel(info_keywords, hotel)`	Get specified infos for given hotel.
`get_infos_by_restaurant(info_keywords, restaurant)`	Get specified infos for given restaurant.
`get_inter_city_transport(from_city, to_city)`	Get all transports between given city pair.
`get_weather_by_city_date(city, date)`	Get weather for given city-date pair.
`submit_result(answer_text)`	Submit the final answer content.

4。评估

4.1。主要结果

我们在 MMLU-Pro (Wang et al., 2024)、GPQA Diamond (Rein et al., 2023)、Human Last Exam (HLE) Text-only (Phan et al., 2025)、LiveCodeBench (2024.08-2025.04)、Codeforces, Aider-Polyglot, AIME 2025, HMMT Feb 2025, HMMT Nov 2025 (Balunović et al., 2025), IMOAnswerBench (Luong et al., 2025), Terminal Bench 2.0, SWE-Verified (OpenAI, 2024b), SWE Multilingual (Yang et al., 2025), BrowseComp (Wei et al., 2025), BrowseCompZh (Zhou et al., 2025), τ^2-bench (Barres et al., 2025), MCP-Universe (Luo et al., 2025), MCP-Mark (EvalSys, 2025), and Tool-Decathlon (Li et al., 2025). Tool-use benchmarks are evaluated using the standard function call format, wherein models are configured to thinking mode. For MCP-Universe (Luo et al., 2025) and MCP-Mark (EvalSys, 2025), we evaluate all models with our internal environment, because the search and playwright environment might be slightly different from the official setting. We set the temperature to 1.0, and the context window to 128K tokens. For math-related tasks such as AIME, HMMT, IMOAnswerBench, and HLE, we eval with the following template: `"{question}\nPlease reason step by step, and put your final answer within \boxed{}."` In the case of HLE, we additionally assessed DeepSeek-V3.2-Thinking using the official template, resulting in a score of 23.9.

Table 2 | Comparison between DeepSeek-V3.2 and closed/open models. For open models, we just compare with models supports thinking in tooluse. Numbers in bold represent the best scores within each model class (open-source and closed-source). The τ^2-Bench result is computed by the average of each category. Regarding BrowseComp, the performance with the context management technique is noted with *.

	Benchmark (Metric)	**Claude-4.5-Sonnet**	**GPT-5 High**	**Gemini-3.0 Pro**	**Kimi-K2 Thinking**	**MiniMax M2**	**DeepSeek-V3.2 Thinking**
English	MMLU-Pro (EM)	88.2	87.5	**90.1**	84.6	82.0	**85.0**
	GPQA Diamond (Pass@1)	83.4	85.7	**91.9**	**84.5**	77.7	82.4
	HLE (Pass@1)	13.7	26.3	**37.7**	23.9	12.5	**25.1**
Code	LiveCodeBench (Pass@1-COT)	64.0	84.5	**90.7**	82.6	83.0	**83.3**
	Codeforces (Rating)	1480	2537	**2708**	-	-	2386
Math	AIME 2025 (Pass@1)	87.0	94.6	**95.0**	**94.5**	78.3	93.1
	HMMT Feb 2025 (Pass@1)	79.2	88.3	**97.5**	89.4	-	**92.5**
	HMMT Nov 2025 (Pass@1)	81.7	89.2	**93.3**	89.2	-	**90.2**
	IMOAnswerBench (Pass@1)	-	76.0	**83.3**	**78.6**	-	78.3
Code Agent	Terminal Bench 2.0 (Acc)	42.8	35.2	**54.2**	35.7	30.0	**46.4**
	SWE Verified (Resolved)	**77.2**	74.9	76.2	71.3	69.4	**73.1**
	SWE Multilingual (Resolved)	**68.0**	55.3	-	61.1	56.5	**70.2**
Search Agent	BrowseComp (Pass@1)	24.1	**54.9**	-	-/60.2*	44.0	**51.4/67.6***
	BrowseCompZh (Pass@1)	42.4	63.0	-	62.3	48.5	**65.0**
	HLE (Pass@1)	32.0	35.2	**45.8**	**44.9**	31.8	40.8
ToolUse	τ^2-Bench(Pass@1)	84.7	80.2	**85.4**	74.3	76.9	**80.3**
	MCP-Universe (Success Rate)	46.5	47.9	**50.7**	35.6	29.4	**45.9**
	MCP-Mark (Pass@1)	33.3	**50.9**	43.1	20.4	24.4	**38.0**
	Tool-Decathlon (Pass@1)	**38.6**	29.0	36.4	17.6	16.0	**35.2**

DeepSeek-V3.2 achieves similar performance with GPT-5-high on reasoning tasks, but is slightly worse than Gemini-3.0-Pro. Compared to K2-Thinking, DeepSeek-V3.2 achieves comparable scores with substantially fewer output tokens, as shown in Table 3. These performance gains can be attributed to the increased computational resources allocated to RL training. Over recent months, we have observed consistent performance improvements correlating with extended RL training budget, which already exceeds 10% of the pre-training cost. We hypothesize that reasoning capabilities could be further enhanced with additional computational budget allocation. Notably, the performance of DeepSeek-V3.2 presented herein is constrained by a length constraint reward model; upon removal of the restriction, we observe further improvement in

力、Aider-Polyglot、AIME 2025、HMMT Feb 2025、HMMT Nov 2025（Balunović 等人，2025）、IMOAswerBench（Luong 等人，2025）、Terminal Bench 2.0、SWE-Verified（OpenAI，2024b）、SWE Multilingual（Yang 等人， 2025）、BrowseComp（Wei 等人，2025）、BrowseCompZh（Zhou 等人，2025）、τ^2-bench（Barres 等人，2025）、MCP-Universe（Luo 等人，2025）、MCP-Mark（EvalSys，2025）和 Tool-Decathlon（Li 等人，2025） 2025）。使用标准函数调用格式评估工具使用基准，其中模型配置为思维模式。对于 MCP-Universe (Luo et al., 2025) 和 MCP-Mark (EvalSys, 2025)，我们使用内部环境评估所有模型，因为搜索和剧作家环境可能与官方设置略有不同。我们将温度设置为 1.0，将上下文窗口设置为 128K 个标记。对于 AIME、HMMT、IMOAswerBench 和 HLE 等数学相关任务，我们使用以下模板进行评估：
`"{question}\nPlease reason step by step, and put your final answer within \boxed{}."` 对于 HLE，我们还使用官方模板评估了 DeepSeek-V3.2-Thinking，得分为 23.9。

表2 | DeepSeek-V3.2 与封闭/开放模型之间的比较。对于开放模型，我们只是与支持工具使用思维的模型进行比较。粗体数字代表每个模型类别（开源和闭源）中的最佳分数。 τ^2-Bench 结果是根据每个类别的平均值计算的。对于 BrowseComp，上下文管理技术的性能用 * 表示。

	Benchmark (Metric)	**Claude-4.5-**	**GPT-5**	**Gemini-3.0**	**Kimi-K2**	**MiniMax**	**DeepSeek-V3.2**
		Sonnet	**High**	**Pro**	**Thinking**	**M2**	**Thinking**
English	MMLU-Pro (EM)	88.2	87.5	**90.1**	84.6	82.0	**85.0**
	GPQA Diamond (Pass@1)	83.4	85.7	**91.9**	**84.5**	77.7	82.4
	HLE (Pass@1)	13.7	26.3	**37.7**	23.9	12.5	**25.1**
Code	LiveCodeBench (Pass@1-COT)	64.0	84.5	**90.7**	82.6	83.0	**83.3**
	Codeforces (Rating)	1480	2537	**2708**	-	-	2386
Math	AIME 2025 (Pass@1)	87.0	94.6	**95.0**	**94.5**	78.3	93.1
	HMMT Feb 2025 (Pass@1)	79.2	88.3	**97.5**	89.4	-	**92.5**
	HMMT Nov 2025 (Pass@1)	81.7	89.2	**93.3**	89.2	-	**90.2**
	IMOAnswerBench (Pass@1)	-	76.0	**83.3**	**78.6**	-	78.3
Code Agent	Terminal Bench 2.0 (Acc)	42.8	35.2	**54.2**	35.7	30.0	**46.4**
	SWE Verified (Resolved)	**77.2**	74.9	76.2	71.3	69.4	**73.1**
	SWE Multilingual (Resolved)	**68.0**	55.3	-	61.1	56.5	**70.2**
Search Agent	BrowseComp (Pass@1)	24.1	**54.9**	-	-/60.2*	44.0	**51.4/67.6***
	BrowseCompZh (Pass@1)	42.4	63.0	-	62.3	48.5	**65.0**
	HLE (Pass@1)	32.0	35.2	**45.8**	**44.9**	31.8	40.8
ToolUse	τ^2-Bench(Pass@1)	84.7	80.2	**85.4**	74.3	76.9	**80.3**
	MCP-Universe (Success Rate)	46.5	47.9	**50.7**	35.6	29.4	**45.9**
	MCP-Mark (Pass@1)	33.3	**50.9**	43.1	20.4	24.4	**38.0**
	Tool-Decathlon (Pass@1)	**38.6**	29.0	36.4	17.6	16.0	**35.2**

DeepSeek-V3.2 在推理任务上实现了与 GPT-5-high 相似的性能，但比 Gemini-3.0-Pro 稍差。与 K2-Thinking 相比，DeepSeek-V3.2 以更少的输出标记实现了可比较的分数，如表 3 所示。这些性能提升可归因于分配给 RL 训练的计算资源的增加。近几个月来，我们观察到与延长 RL 训练预算相关的持续性能改进，该预算已超过预训练成本的 10%。我们假设通过额外的计算预算分配可以进一步增强推理能力。值得注意的是，本文提出的 DeepSeek-V3.2 的性能受到长度约束奖励模型的约束；取消限制后，我们观察到进一步改善

model performance, as detailed in Section 4.2.

In code agent evaluations, DeepSeek-V3.2 significantly outperforms open-source LLMs on both SWE-bench Verified and Terminal Bench 2.0, demonstrating its potential within real-world coding workflows. Regarding Terminal Bench 2.0, as previously noted, our context management strategy for the 'thinking mode' is currently incompatible with Terminus; consequently, the reported score of 46.4 was achieved using the Claude Code framework. We also evaluated DeepSeek-V3.2 with Terminus in non-thinking mode, yielding a score of 39.3. For SWE-bench Verified, the primary score was obtained using our internal framework. Robustness tests across other settings—including the Claude Code and RooCode frameworks, as well as non-thinking mode—produced consistent results, ranging from 72 to 74.

For the search agent evaluation, we assess our models using a standard commercial search API. Since DeepSeek-V3.2 supports a maximum context length of only 128K, approximately 20%+ of the test cases exceed this limit. To address this, we employ a context management method to derive the final score. For reference, the score is 51.4 without context management. Further details are provided in Section 4.4.

On tool-use benchmarks, DeepSeek-V3.2 substantially narrows the performance gap between open-source and closed-source LLMs, though it remains below frontier models. For τ^2-bench, we employ the model itself as the user agent, achieving final category scores of 63.8 (Airline), 81.1 (Retail), and 96.2 (Telecom). For the MCP benchmarks, we employ the function calling format and place tool outputs within messages designated with the 'tool' role, rather than the 'user' role. During our testing, we observed that DeepSeek-V3.2 frequently engages in redundant self-verification, generating excessively long trajectories. This tendency often causes the context length to exceed the 128K limit, particularly in tasks such as MCP-Mark GitHub and Playwright evaluation. Consequently, this phenomenon hinders the final performance of DeepSeek-V3.2. However, integrating context management strategies can further enhance performance. We identify this as a direction for future work and a practical consideration for users. Even if DeepSeek-V3.2 suffers from the issue, it still significantly outperforms existing open models. Notably, since the environments and toolsets employed in these benchmarks were not encountered during RL training, the observed improvements demonstrate DeepSeek-V3.2's capacity to generalize its reasoning strategies to out-of-domain agentic scenarios. The evaluation of non-thinking model in the agent scenario is shown in Appendix Table 9.

4.2. Results of DeepSeek-V3.2-Speciale

Table 3 demonstrates that DeepSeek-V3.2-Speciale achieves superior performance by leveraging increased reasoning tokens, surpassing the state-of-the-art Gemini-3.0-Pro across multiple benchmarks. Remarkably, as shown in Table 4, this general-purpose model attains gold-medal level performance in the 2025 International Olympiad in Informatics (IOI) and the ICPC World Finals (ICPC WF) without targeted training. Furthermore, by incorporating techniques from Shao et al. (2025), the model excels in complex proof tasks, reaching gold-medal thresholds in the 2025 International Mathematical Olympiad (IMO) and China Mathematical Olympiad (CMO)[5]. Detailed evaluation protocols are provided in Appendix D.

However, the token efficiency of DeepSeek-V3.2-Speciale remains significantly inferior to that of Gemini-3.0-Pro. To mitigate deployment costs and latency, we imposed stricter token constraints during the training of the official DeepSeek-V3.2, aiming to optimize the trade-off

[5]We evaluated the English version of CMO 2025. The IMO 2025 and CMO 2025 problems, together with the inference code, can be found at: `https://github.com/deepseek-ai/DeepSeek-Math-V2`.

模型性能，详见 4.2 节。

在代码代理评估中，DeepSeek-V3.2 在 SWE-bench Verified 和 Terminal Bench 2.0 上均显着优于开源 LLM，展示了其在实际编码工作流程中的潜力。关于 Terminal Bench 2.0，如前所述，我们的“思维模式”上下文管理策略目前与 Terminus 不兼容；因此，使用 Claude Code 框架获得了 46.4 分的报告分数。我们还在非思考模式下使用 Terminus 评估了 DeepSeek-V3.2，得分为 39.3。对于 SWE-bench Verified，主要分数是使用我们的内部框架获得的。跨其他设置（包括 Claude Code 和 RooCode 框架以及非思考模式）的稳健性测试产生了一致的结果，范围从 72 到 74。

对于搜索代理评估，我们使用标准商业搜索 API 来评估我们的模型。由于 DeepSeek-V3.2 支持的最大上下文长度仅为 128K，因此大约 20%+ 的测试用例超出了此限制。为了解决这个问题，我们采用上下文管理方法来得出最终分数。作为参考，没有上下文管理的情况下得分为 51.4。第 4.4 节提供了更多详细信息。

在工具使用基准测试中，DeepSeek-V3.2 大大缩小了开源和闭源 LLM 之间的性能差距，尽管它仍然低于前沿模型。对于 τ^2-bench，我们使用模型本身作为用户代理，最终类别得分为 63.8（航空）、81.1（零售）和 96.2（电信）。对于 MCP 基准测试，我们采用函数调用格式，并将工具输出放置在指定为“工具”角色而不是“用户”角色的消息中。在我们的测试过程中，我们观察到 DeepSeek-V3.2 经常进行冗余的自我验证，生成过长的轨迹。这种趋势通常会导致上下文长度超过 128K 限制，特别是在 MCP-Mark GitHub 和 Playwright 评估等任务中。因此，这种现象阻碍了DeepSeek-V3.2的最终性能。然而，集成上下文管理策略可以进一步提高性能。我们将此作为今后工作的方向，也是对用户的实际考虑。即使 DeepSeek-V3.2 遇到这个问题，它仍然明显优于现有的开放模型。值得注意的是，由于在 RL 训练期间没有遇到这些基准测试中使用的环境和工具集，因此观察到的改进表明 DeepSeek-V3.2 能够将其推理策略推广到域外代理场景。 Agent场景下非思维模型的评价如附表9所示。

4.2. DeepSeek-V3.2-Speciale 的结果

表 3 表明 DeepSeek-V3.2-Speciale 通过利用增加的推理令牌实现了卓越的性能，在多个基准测试中超越了最先进的 Gemini-3.0-Pro。值得注意的是，如表 4 所示，该通用模型在 2025 年国际信息学奥林匹克（IOI）和 ICPC 世界总决赛（ICPC WF）中无需进行针对性训练即可获得金牌水平的表现。此外，通过结合 Shao 等人的技术。 (2025)，该模型在复杂的证明任务中表现出色，在 2025 年国际数学奥林匹克 (IMO) 和中国数学奥林匹克 (CMO)[5] 中达到了金牌门槛。附录 D 提供了详细的评估方案。

然而，DeepSeek-V3.2-Speciale 的代币效率仍然明显逊色于 Gemini-3.0-Pro。为了降低部署成本和延迟，我们在官方 DeepSeek-V3.2 的训练过程中施加了更严格的代币约束，旨在优化权衡

[5]We evaluated the English version of CMO 2025. The IMO 2025 and CMO 2025 problems, together with the inference code, can be found at: `https://github.com/deepseek-ai/DeepSeek-Math-V2`.

Table 3 | Benchmark performance and efficiency of reasoning models. For each benchmark, cells show accuracy and output token count (in thousands). The highest accuracy per benchmark is in bold; the second-highest is underlined.

Benchmark	**GPT-5 High**	**Gemini-3.0 Pro**	**Kimi-K2 Thinking**	**DeepSeek-V3.2 Thinking**	**DeepSeek-V3.2 Speciale**
AIME 2025 (Pass@1)	94.6 (13k)	<u>95.0</u> (15k)	94.5 (24k)	93.1 (16k)	**96.0** (23k)
HMMT Feb 2025 (Pass@1)	88.3 (16k)	<u>97.5</u> (16k)	89.4 (31k)	92.5 (19k)	**99.2** (27k)
HMMT Nov 2025 (Pass@1)	89.2 (20k)	<u>93.3</u> (15k)	89.2 (29k)	90.2 (18k)	**94.4** (25k)
IMOAnswerBench (Pass@1)	76.0 (31k)	<u>83.3</u> (18k)	78.6 (37k)	78.3 (27k)	**84.5** (45k)
LiveCodeBench (Pass@1-COT)	84.5 (13k)	**90.7** (13k)	82.6 (29k)	83.3 (16k)	<u>88.7</u> (27k)
CodeForces (Rating)	2537 (29k)	**2708** (22k)	-	2386 (42k)	<u>2701</u> (77k)
GPQA Diamond (Pass@1)	<u>85.7</u> (8k)	**91.9** (8k)	84.5 (12k)	82.4 (7k)	<u>85.7</u> (16k)
HLE (Pass@1)	26.3 (15k)	**37.7** (15k)	23.9 (24k)	25.1 (21k)	<u>30.6</u> (35k)

between performance and cost. We believe that token efficiency remains a critical area for future investigation.

Table 4 | Performance of DeepSeek-V3.2-Speciale in top-tier mathematics and coding competitions. For ICPC WF 2025, we report the number of submissions for each successfully solved problem. DeepSeek-V3.2-Speciale ranked 2nd in ICPC WF 2025 and 10th in IOI 2025.

Competition	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**Overall**	**Medal**
IMO 2025	7	7	7	7	7	0	35/42	Gold
CMO 2025	18	18	9	21	18	18	102/126	Gold
IOI 2025	100	82	72	100	55	83	492/600	Gold

Competition	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**Overall**	**Medal**
ICPC WF 2025	3	-	1	1	2	2	-	1	1	1	1	1	10/12	Gold

4.3. Synthesis Agentic Tasks

In this section, we perform ablation experiments to study the effect of synthetic agentic tasks. We focus on two questions. First, are synthetic tasks sufficiently challenging for reinforcement learning? Second, how well do these synthetic tasks generalize, i.e., can they transfer to different downstream tasks or real-world environments?

To address the first question, we randomly sample 50 instances from the general synthesized agentic tasks and evaluate both the model used for synthesis and frontier closed-source LLMs. As shown in Table 5, DeepSeek-V3.2-Exp attains an accuracy of only 12%, while frontier closed-source models achieve at most 62%. These results indicate that the synthetic data include agentic tasks that are challenging for both DeepSeek-V3.2-Exp and frontier closed-source models.

To investigate whether RL on synthetic data can generalize to different tasks or real-world environments, we apply RL to the SFT checkpoint of DeepSeek-V3.2 (denoted DeepSeek-V3.2-SFT). To exclude the effects of long CoT and other RL data, we conduct RL only on synthetic agentic tasks in non-thinking mode. We then compare the model with DeepSeek-V3.2-SFT and DeepSeek-V3.2-Exp, where DeepSeek-V3.2-Exp is trained with RL only in search and code environments. As shown in Figure 5, large-scale RL on synthetic data yields substantial improve-

表 3 | 推理模型的基准性能和效率。对于每个基准，单元格显示准确性和输出令牌计数（以千为单位）。每个基准的最高准确度以粗体显示；第二高的有下划线。

Benchmark	**GPT-5 High**	**Gemini-3.0 Pro**	**Kimi-K2 Thinking**	**DeepSeek-V3.2 Thinking**	**DeepSeek-V3.2 Speciale**
AIME 2025 (Pass@1)	94.6 (13k)	95.0 (15k)	94.5 (24k)	93.1 (16k)	**96.0** (23k)
HMMT Feb 2025 (Pass@1)	88.3 (16k)	97.5 (16k)	89.4 (31k)	92.5 (19k)	**99.2** (27k)
HMMT Nov 2025 (Pass@1)	89.2 (20k)	93.3 (15k)	89.2 (29k)	90.2 (18k)	**94.4** (25k)
IMOAnswerBench (Pass@1)	76.0 (31k)	83.3 (18k)	78.6 (37k)	78.3 (27k)	**84.5** (45k)
LiveCodeBench (Pass@1-COT)	84.5 (13k)	**90.7** (13k)	82.6 (29k)	83.3 (16k)	88.7 (27k)
CodeForces (Rating)	2537 (29k)	**2708** (22k)	-	2386 (42k)	2701 (77k)
GPQA Diamond (Pass@1)	85.7 (8k)	**91.9** (8k)	84.5 (12k)	82.4 (7k)	85.7 (16k)
HLE (Pass@1)	26.3 (15k)	**37.7** (15k)	23.9 (24k)	25.1 (21k)	30.6 (35k)

性能和成本之间。我们认为代币效率仍然是未来研究的关键领域。

表 4 | DeepSeek-V3.2-Speciale 在顶级数学和编码竞赛中的表现。对于 ICPC WF 2025，我们报告每个成功解决问题的提交数量。 DeepSeek-V3.2-Speciale 在 ICPC WF 2025 中排名第二，在 IOI 2025 中排名第十。

Competition	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**Overall**	**Medal**
IMO 2025	7	7	7	7	7	0	35/42	Gold
CMO 2025	18	18	9	21	18	18	102/126	Gold
IOI 2025	100	82	72	100	55	83	492/600	Gold

Competition	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**Overall**	**Medal**
ICPC WF 2025	3	-	1	1	2	2	-	1	1	1	1	1	10/12	Gold

4.3.综合代理任务

在本节中，我们进行消融实验来研究合成代理任务的效果。我们关注两个问题。首先，综合任务对于强化学习来说是否足够具有挑战性？其次，这些综合任务的泛化能力如何，即它们能否转移到不同的下游任务或现实环境中？

为了解决第一个问题，我们从一般综合代理任务中随机抽取 50 个实例，并评估用于综合的模型和前沿闭源 LLM。如表5所示，DeepSeek-V3.2-Exp的准确率仅为12%，而前沿闭源模型最多达到62%。这些结果表明，合成数据包括对 DeepSeek-V3.2-Exp 和前沿闭源模型都具有挑战性的代理任务。

为了研究合成数据上的强化学习是否可以推广到不同的任务或现实环境，我们将强化学习应用于 DeepSeek-V3.2 的 SFT 检查点（表示为 DeepSeek-V3.2-SFT）。为了排除长 CoT 和其他 RL 数据的影响，我们仅在非思维模式下对合成代理任务进行 RL。然后，我们将该模型与 DeepSeek-V3.2-SFT 和 DeepSeek-V3.2-Exp 进行比较，其中 DeepSeek-V3.2-Exp 仅在搜索和代码环境中使用 RL 进行训练。如图 5 所示，对合成数据进行大规模 RL 产生了显着的改进：

Table 5 | Accuracy of general synthesized tasks on different models.

Pass@K	DeepSeek-v3.2-Exp	Sonnet-4.5	Gemini-3.0 Pro	GPT-5-Thinking
1	12%	34%	51%	62%
2	18%	47%	65%	75%
4	26%	62%	74%	82%

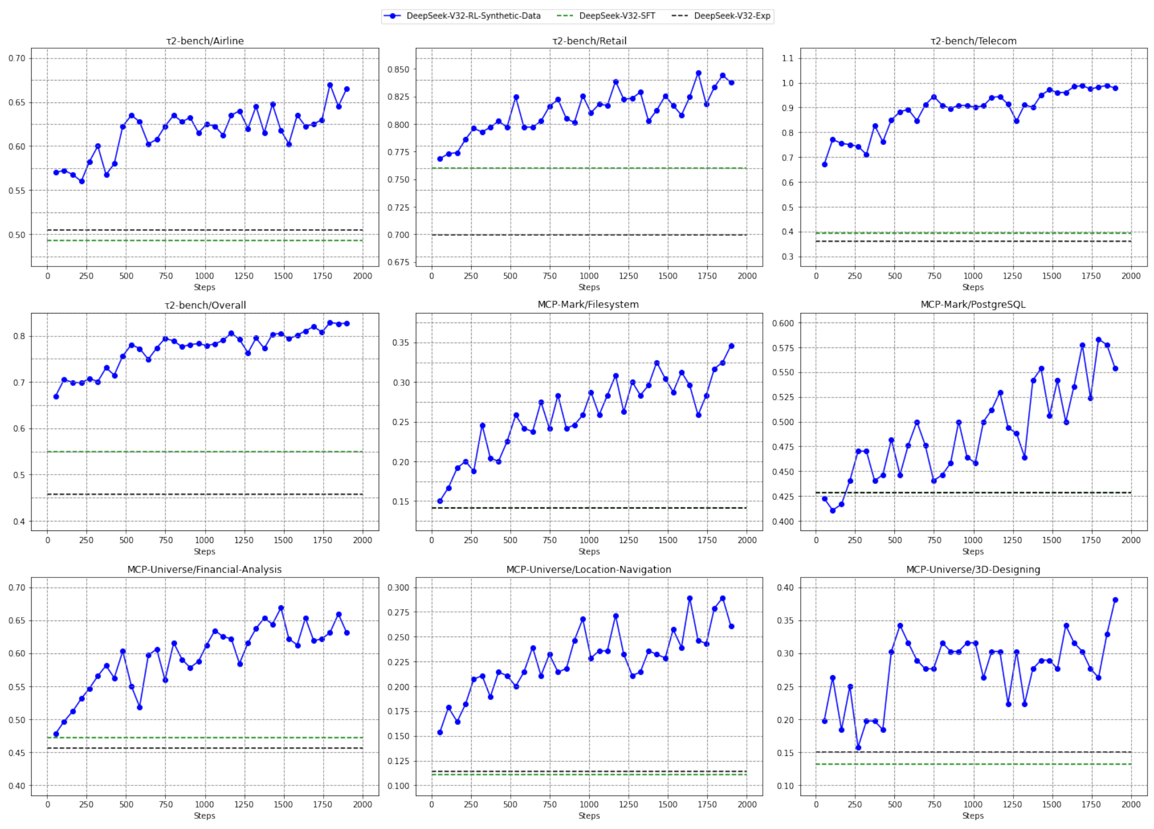


Figure 5 | RL training of DeepSeek-V3.2-SFT using exclusively synthetic general agent data.

ments over DeepSeek-V3.2-SFT on Tau2Bench, MCP-Mark, and MCP-Universe benchmarks. In contrast, restricting RL to code and search scenarios does not improve performance on these benchmarks, further highlighting the potential of synthetic data.

4.4. Context Management of Search Agent

Even with extended context windows such as 128k, agentic workflows, particularly in search-based scenarios, frequently encounter maximum length limitations that prematurely truncate the reasoning process. This bottleneck inhibits the full realization of test-time compute potential. To address this, we introduce context management employing simple strategies to extend token budgets at test time， when the token usage exceeds 80% of the context window length. These strategies include (1) **Summary**, which summarizes the overflowed trajectory and re-initiates the rollout; (2) **Discard-75%**, which discards the first 75% tool call history in the trajectory to free up spaces; (3) **Discard-all**, which resets the context by discarding all previous tool call history (similar to the new context tool (Anthropic, 2025a)). For comparison, we also implement a parallel scaling baseline, **Parallel-fewest-step**, which samples N independent trajectories and

表 5 | 一般综合任务在不同模型上的准确率。

Pass@K	DeepSeek-v3.2-Exp	Sonnet-4.5	Gemini-3.0 Pro	GPT-5-Thinking
1	12%	34%	51%	62%
2	18%	47%	65%	75%
4	26%	62%	74%	82%

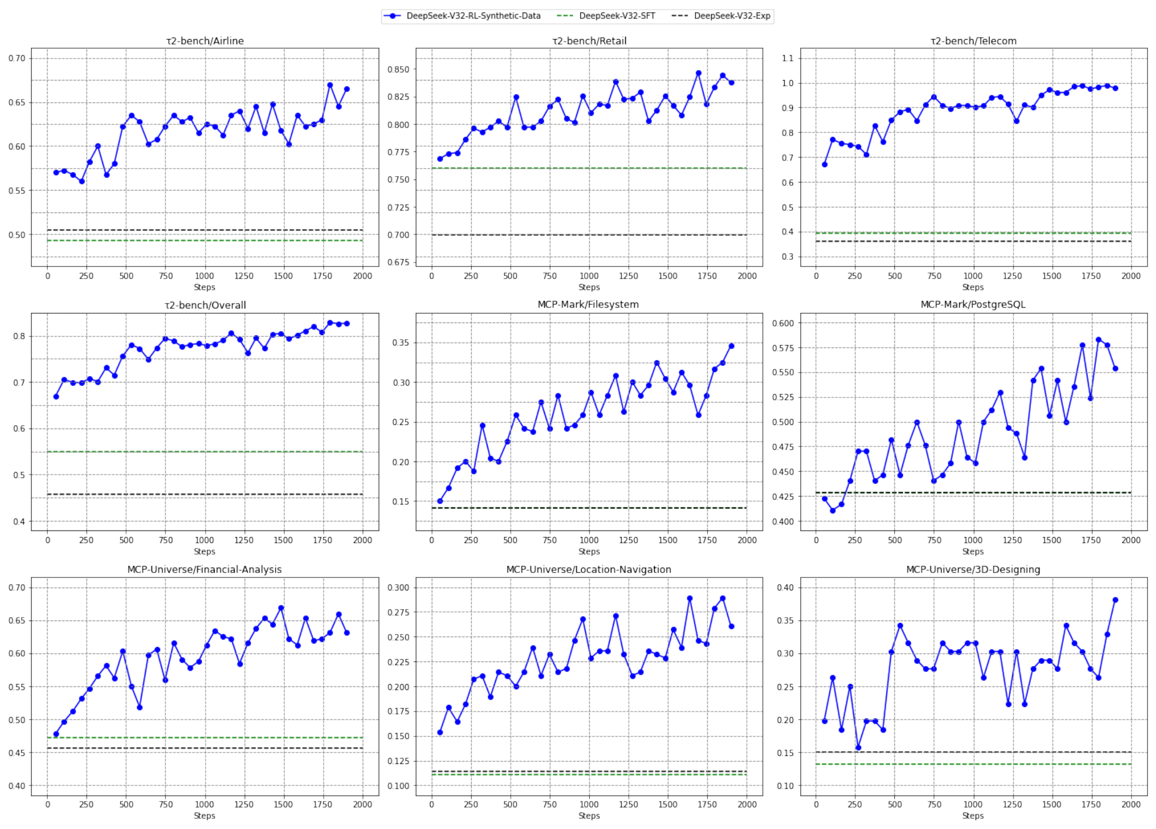


图 5 使用专门合成的通用代理数据对 DeepSeek-V3.2-SFT 进行 | RL 训练。

Tau2Bench、MCP-Mark 和 MCP-Universe 基准测试中 DeepSeek-V3.2-SFT 的评价。相比之下，将强化学习限制在代码和搜索场景中并不能提高这些基准的性能，这进一步凸显了合成数据的潜力。

4.4.搜索代理的上下文管理

即使使用扩展的上下文窗口（例如 128k），代理工作流程（尤其是在基于搜索的场景中）也经常遇到最大长度限制，从而过早地截断推理过程。这一瓶颈阻碍了测试时计算潜力的充分实现。为了解决这个问题，我们引入了上下文管理，当令牌使用量超过上下文窗口长度的 80% 时，使用简单的策略来扩展测试时的令牌预算。这些策略包括（1）Summary，总结溢出的轨迹并重新启动rollout；（2）Discard-75%，丢弃轨迹中前75%的工具调用历史，以释放空间；（3）Discard-all，通过丢弃所有以前的工具调用历史来重置上下文（类似于新的上下文工具（Anthropic，2025a））。为了进行比较，我们还实现了一个并行缩放基线，Parallel-feest-step，它对 N 个独立的轨迹进行采样，并

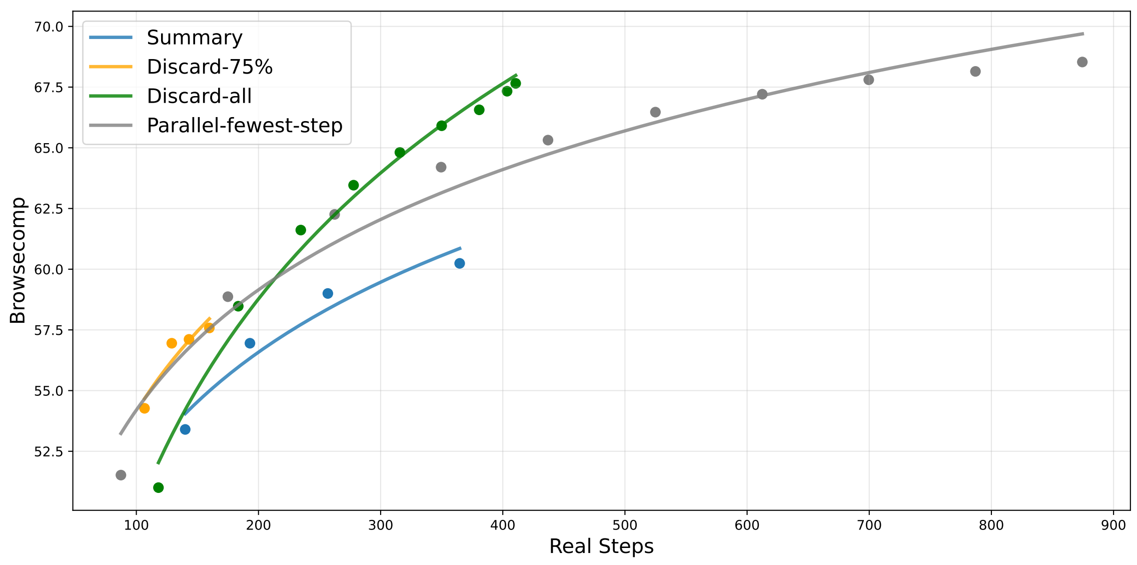


Figure 6 | Accuracy of Browsecomp with different test-time compute expansion strategies.

selects the trajectory with the fewest steps.

We evaluate these strategies on the BrowseComp benchmark (Wei et al., 2025). As illustrated in Figure 6, under varying compute budgets, context management leads to significant performance gains by allowing the model to scale up test-time compute, providing more space to perform additional execution steps. For example, Summary extends the average steps from 140 to 364, improving performance from 53.4 to 60.2. However, its overall efficiency is relatively low. Despite its simplicity, Discard-all performs well in both efficiency and scalability, achieving a score of 67.6, comparable to parallel scaling while using significantly fewer steps.

In summary, test-time compute can be scaled either serially through context management or in parallel, both effectively extending the model's problem-solving capacity. However, different strategies exhibit varying efficiency and scalability. Thus, it is crucial to account for actual compute costs when benchmarking model performance. Meanwhile, finding the optimal combination of serial and parallel scaling to maximize both efficiency and scalability remains a crucial direction for future work.

5. Conclusion, Limitation, and Future Work

In this work, we introduced DeepSeek-V3.2, a framework that effectively bridges the gap between computational efficiency and advanced reasoning capabilities. Using DSA, we addressed critical computation complexity without sacrificing long-context performance. By increasing computational budget, DeepSeek-V3.2 achieves comparable performance with GPT-5 on reasoning benchmarks. Finally, the integration of our large-scale agentic task synthesis pipeline significantly enhances tool-use proficiency, unlocking new possibilities for robust and generalizable AI agents with open LLM. Furthermore, our high-compute variant, DeepSeek-V3.2-Speciale, validated by gold-medal achievements in the IMO and IOI, sets a milestone for open LLMs.

Despite these achievements, we acknowledge certain limitations when compared to frontier closed-source models such as Gemini-3.0-Pro. First, due to fewer total training FLOPs, the breadth of world knowledge in DeepSeek-V3.2 still lags behind that of leading proprietary

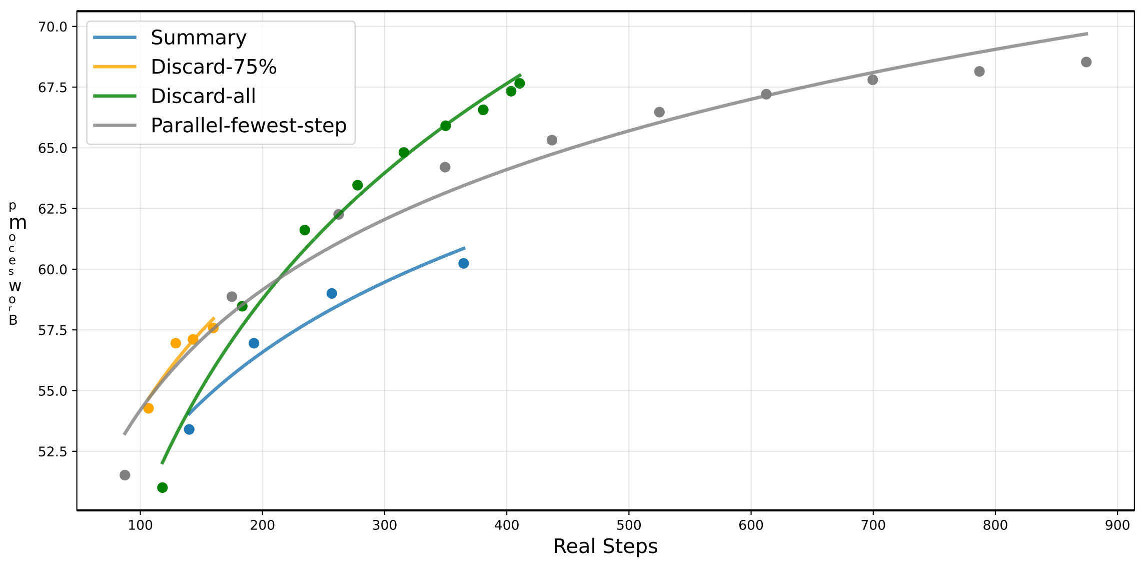


图 6 | Browsecomp 在不同测试时计算扩展策略下的准确性。

选择步数最少的轨迹。

我们在 BrowseComp 基准上评估这些策略（Wei 等人，2025）。如图 6 所示，在不同的计算预算下，上下文管理允许模型扩展测试时计算，提供更多空间来执行额外的执行步骤，从而显着提高性能。例如，Summary 将平均步数从 140 扩展到 364，将性能从 53.4 提高到 60.2。但其整体效率较低。尽管很简单，Discard-all 在效率和可扩展性方面都表现良好，得分为 67.6，与并行扩展相当，同时使用的步骤少得多。

总之，测试时计算可以通过上下文管理串行或并行扩展，两者都有效扩展了模型解决问题的能力。然而，不同的策略表现出不同的效率和可扩展性。因此，在对模型性能进行基准测试时考虑实际计算成本至关重要。同时，找到串行和并行扩展的最佳组合以最大限度地提高效率和可扩展性仍然是未来工作的关键方向。

5。结论，限制和未来工作

在这项工作中，我们引入了 DeepSeek-V3.2，这是一个有效弥合计算效率和高级推理能力之间差距的框架。使用 DSA，我们在不牺牲长上下文性能的情况下解决了关键的计算复杂性。通过增加计算预算，DeepSeek-V3.2 在推理基准上实现了与 GPT-5 相当的性能。最后，我们大规模代理任务合成管道的集成显着提高了工具使用熟练程度，为具有开放法学硕士的强大且可通用的人工智能代理释放了新的可能性。此外，我们的高计算版本 DeepSeek-V3.2-Speciale 经过 IMO 和 IOI 金牌成就的验证，为开放式法学硕士树立了里程碑。

尽管取得了这些成就，但我们承认与 Gemini-3.0-Pro 等前沿闭源模型相比存在某些局限性。首先，由于总训练 FLOP 较少，DeepSeek-V3.2 的世界知识广度仍然落后于领先的专有技术

models. We plan to address this knowledge gap in future iterations by scaling up the pre-training compute. Second, token efficiency remains a challenge; DeepSeek-V3.2 typically requires longer generation trajectories (i.e., more tokens) to match the output quality of models like Gemini-3.0-Pro. Future work will focus on optimizing the intelligence density of the model's reasoning chains to improve efficiency. Third, solving complex tasks is still inferior to frontier models, motivating us to further refine our foundation model and post-training recipe.

References

Anthropic. System card: Claude opus 4.5, 2025a. URL https://assets.anthropic.com/m/64823ba7485345a7/Claude-Opus-4-5-System-Card.pdf.

Anthropic. Introducing claude sonnet 4.5, 2025b. URL https://www.anthropic.com/news/claude-sonnet-4-5l.

M. Balunović, J. Dekoninck, I. Petrov, N. Jovanović, and M. Vechev. Matharena: Evaluating llms on uncontaminated math competitions. Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmark, 2025.

V. Barres, H. Dong, S. Ray, X. Si, and K. Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261, 2025a.

G. DeepMind. Gemini 3 pro model card, 2025b. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. CoRR, abs/2405.04434, 2024. doi: 10.48550/ARXIV.2405.04434. URL https://doi.org/10.48550/arXiv.2405.04434.

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. Nature, 645(8081):633–638, 2025.

EvalSys. Mcpmark leaderboard, 2025. URL https://mcpmark.ai/leaderboard.

J. Li, W. Zhao, J. Zhao, W. Zeng, H. Wu, X. Wang, R. Ge, Y. Cao, Y. Huang, W. Liu, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. arXiv preprint arXiv:2510.25726, 2025.

Z. Luo, Z. Shen, W. Yang, Z. Zhao, P. Jwalapuram, A. Saha, D. Sahoo, S. Savarese, C. Xiong, and J. Li. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. arXiv preprint arXiv:2508.14704, 2025.

T. Luong, D. Hwang, H. H. Nguyen, G. Ghiasi, Y. Chervonyi, I. Seo, J. Kim, G. Bingham, J. Lee, S. Mishra, A. Zhai, C. H. Hu, H. Michalewski, J. Kim, J. Ahn, J. Bae, X. Song, T. H. Trinh, Q. V. Le, and J. Jung. Towards robust mathematical reasoning. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025. URL https://aclanthology.org/2025.emnlp-main.1794/.

模型。我们计划通过扩大预训练计算来在未来的迭代中解决这一知识差距。其次，代币效率仍然是一个挑战； DeepSeek-V3.2 通常需要更长的生成轨迹（即更多令牌）来匹配 Gemini-3.0-Pro 等模型的输出质量。未来的工作将重点优化模型推理链的智能密度，以提高效率。第三，解决复杂任务仍然不如前沿模型，这促使我们进一步完善我们的基础模型和后训练配方。

参考

人为的。系统卡：Claude opus 4.5，2025a。网址 `https://assets.anthropic.com/m/64823ba7485345a7/Claude-Opus-4-5-System-Card.pdf`。

人为的。介绍克劳德十四行诗 4.5，2025b。网址`https://www.anthropic.com/news/claude-sonnet-4-5l`。

M. Balunović、J. Dekoninck、I. Petrov、N. Jovanović 和 M. Vechev。 Matharena：在未受污染的数学竞赛中评估 llms。神经信息处理系统数据集和基准跟踪论文集，2025 年。

V. Barres、H. Dong、S. Ray、X. Si 和 K. Narasimhan。 τ^2-bench：评估双控制环境中的会话代理，2025 年。URL `https://arxiv.org/abs/2506.07982`。

深心。 Gemini 2.5：利用高级推理、多模态、长上下文和下一代代理功能推动前沿。 arXiv 预印本 arXiv:2507.06261, 2025a。

G.DeepMind。 Gemini 3 Pro 型号卡，2025b。网址 `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf`。

DeepSeek-AI。 Deepseek-v2：强大、经济且高效的专家混合语言模型。 CoRR，abs/2405.04434，2024。doi：10.48550/ARXIV.2405.04434。网址 `https://doi.org/10.48550/arXiv.2405.04434`。

DeepSeek-AI。 Deepseek-v3 技术报告，2024 年。URL `https://arxiv.org/abs/2412.19437`。

DeepSeek-ai。 DeepSeek-R1通过加强学习激励LLM中的推理。自然，645（8081）：633–638，2025。

评估系统。 Mcpmark 排行榜，2025 年。URL `https://mcpmark.ai/leaderboard`。

J. Li，W.Zhao，J.Zhao，W.Zeng，H.Wu，X.Wang，R.Ge，Y.Cao，Y.Huang，W.Liu，等。工具十项全能：对语言代理进行基准测试，以执行多样化、现实且长期的任务。 arXiv 预印本 arXiv: 2510.25726, 2025。

Z. 罗、Z. 沉、W. 杨、Z. 赵、P. Jwalapuram、A. Saha、D. Sahoo、S. Savarese、C. Xiong 和 J. Li。 Mcp-universe：使用真实世界模型上下文协议服务器对大型语言模型进行基准测试。 arXiv 预印本 arXiv:2508.14704, 2025。

T. Luong、D. Hwang、H. H. Nguyen、G. Ghiasi、Y. Chervonyi、I. Seo、J. Kim、G. Bingham、J. Lee、S. Mishra、A. Zhai、C. H. Hu、H. Michalewski、J. Kim、J. Ahn、J. Bae、X. Song、T. H. Trinh、Q. V. Le 和 J. Jung。迈向稳健的数学推理。 2025 年自然语言处理经验方法会议论文集，2025 年。URL `https://aclanthology.org/2025.emnlp-main.1794/`。

MiniMax. https://www.minimax.io/news/minimax-m2, 2025. URL `https://www.minimax.io/news/minimax-m2`.

MoonShot. Introducing kimi k2 thinking, 2025. URL `https://moonshotai.github.io/Kimi-K2/thinking.html`.

OpenAI. Learning to reason with llms, 2024a. URL `https://openai.com/index/learning-to-reason-with-llms/`.

OpenAI. Introducing SWE-bench verified we're releasing a human-validated subset of swe-bench that more, 2024b. URL `https://openai.com/index/introducing-swe-bench-verified/`.

OpenAI. Introducing gpt-5, 2025. URL `https://openai.com/index/introducing-gpt-5/`.

L. Phan, A. Gatti, Z. Han, N. Li, J. Hu, H. Zhang, C. B. C. Zhang, M. Shaaban, J. Ling, S. Shi, et al. Humanity's last exam. arXiv preprint arXiv:2501.14249, 2025.

D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. arXiv preprint arXiv:2311.12022, 2023.

J. Schulman. Approximating KL divergence, 2020. URL `http://joschu.net/blog/kl-approx.html`.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. CoRR, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL `https://doi.org/10.48550/arXiv.2402.03300`.

Z. Shao, Y. Luo, C. Lu, Z. Ren, J. Hu, T. Ye, Z. Gou, S. Ma, and X. Zhang. Deepseekmath-v2: Towards self-verifiable mathematical reasoning, 2025.

N. Shazeer. Fast transformer decoding: One write-head is all you need. CoRR, abs/1911.02150, 2019. URL `http://arxiv.org/abs/1911.02150`.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. pages 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, and W. Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. CoRR, abs/2406.01574, 2024. URL `https://doi.org/10.48550/arXiv.2406.01574`.

J. Wei, Z. Sun, S. Papay, S. McKinney, J. Han, I. Fulford, H. W. Chung, A. T. Passos, W. Fedus, and A. Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. arXiv preprint arXiv:2504.12516, 2025.

J. Yang, K. Lieret, C. E. Jimenez, A. Wettig, K. Khandpur, Y. Zhang, B. Hui, O. Press, L. Schmidt, and D. Yang. Swe-smith: Scaling data for software engineering agents, 2025. URL `https://arxiv.org/abs/2504.21798`.

最小最大。 https://www.minimax.io/news/minimax-m2，2025。URL https://www.minimax.io/news/minimax-m2。

登月。介绍 kimi k2 思维，2025 年。URL https://moonshotai.github.io/Kimi-K2/thinking.html。开放人工智能。学习 llms 推理，2024a。网址 https://openai.com/index/learnin g-to-reason-with-llms/。开放人工智能。引入 SWE-bench 验证后，我们将在 2024b 发布经过人工验证的 swe-bench 子集。网址 https://openai.com/index/introducing-swe-bench -verified/。

开放人工智能。引入 gpt-5，2025。URL
https://openai.com/index/introducing-gpt-5 /。

L.Phan、A.Gatti、Z.Han、N.Li、J.Hu、H.Zhang、C.B.C.Zhang、M.Shaaban、J.Ling、S.Shi 等。人类最后的考试。 arXiv 预印本 arXiv:2501.14249, 2025。

D. Rein，B。L。Hou，A。C。Stickland，J。Petty，R。Y。Pang，J。Dirani，J。Michael和S. R. Bowman。 GPQA：研究生级的Google-Progron-Prover问答基准。 Arxiv预印型ARXIV：2311.12022，2023。
J·舒尔曼。近似 KL 散度，2020。URL http://joschu.net/blog/kl-app rox.html。

Z. Shao、P. Wang、Q. Zhu、R. Xu、J. Song、M. 张、Y. K. Li、Y. Wu 和 D.Guo。 Deepseek- math：突破开放语言模型中数学推理的极限。 CoRR，abs/2402.03300，2024。doi：10.48550/ARXIV.2402.03300。网址 https://doi.org/10 .48550/arXiv.2402.03300。

Z. Shao、Y. Luo、C. Lu、Z. Ren、J. Hu、T. Ye、Z. Gou、S. Ma 和 X. Zhang。 Deepseekmath-v2：迈向可自我验证的数学推理，2025 年。

N.沙泽尔。快速变压器解码：您只需要一个写入头。 CoRR，abs/1911.02150，2019。URL http://arxiv.org/abs/1911.02150。

A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、L. Kaiser 和 I. Polosukhin。您所需要的就是关注。第 5998–6008 页，2017 年。网址 https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html。

Y. Wang、X. Ma、G. 张、Y. Ni、A. Chandra、S.郭、W. Ren、A. Arulraj、X. He、Z. Jiang、T. Li、M. Ku、K. Wang、A. Zhuang、R. Fan、X. Yue 和 W. Chen。 Mmlu-pro：更强大、更具挑战性的多任务语言理解基准。 CoRR，abs/2406.01574，2024。URL
https://doi.org/10.48550/arXiv.2406.01574。

J. Wei、Z. Sun、S. Papay、S. McKinney、J. Han、I. Fulford、H. W. Chung、A. T. Passos、W. Fedus 和 A. Glaese。 Browsecomp：浏览代理的简单但具有挑战性的基准。 arXiv 预印本 arXiv:2504.12516, 2025。

J. Yang、K. Lieret、C. E. Jimenez、A. Wettig、K. Khandpur、Y. 张、B. Hui、O. Press、L. Schmidt 和 D. Yang。 Swe-smith：软件工程代理的扩展数据，2025 年。URL https://arxiv.org/abs/2504.21798。

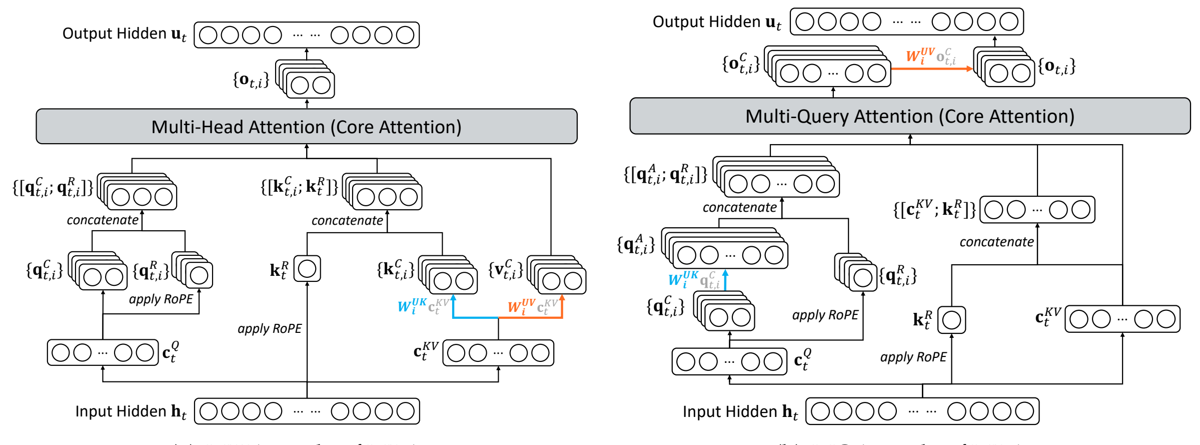
J. Yuan, H. Gao, D. Dai, J. Luo, L. Zhao, Z. Zhang, Z. Xie, Y. Wei, L. Wang, Z. Xiao, Y. Wang, C. Ruan, M. Zhang, W. Liang, and W. Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, pages 23078–23097. Association for Computational Linguistics, 2025. URL `https://aclanthology.org/2025.acl-long.1126/`.

ZhiPu-AI. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. arXiv preprint arXiv:2508.06471, 2025.

P. Zhou, B. Leon, X. Ying, C. Zhang, Y. Shao, Q. Ye, D. Chong, Z. Jin, C. Xie, M. Cao, et al. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. arXiv preprint arXiv:2504.19314, 2025.

Appendices

A. MHA and MQA Modes of MLA


Output Hidden $\mathbf{u}_t$
$\{\mathbf{o}_{t,i}\}$
Multi-Head Attention (Core Attention)
$\{[\mathbf{q}^C_{t,i};\mathbf{q}^R_{t,i}]\}$
$\{[\mathbf{k}^C_{t,i};\mathbf{k}^R_t]\}$
concatenate
$\{\mathbf{q}^C_{t,i}\}$
$\{\mathbf{q}^R_{t,i}\}$
apply RoPE
$\mathbf{k}^R_t$
$\{\mathbf{k}^C_{t,i}\}$
$\{\mathbf{v}^C_{t,i}\}$
$W^{UK}_i \mathbf{c}^{KV}_t$
$W^{UV}_i \mathbf{c}^{KV}_t$
$\mathbf{c}^Q_t$
$\mathbf{c}^{KV}_t$
Input Hidden $\mathbf{h}_t$
Output Hidden $\mathbf{u}_t$
$\{\mathbf{o}^C_{t,i}\}$
$W^{UV}_i \mathbf{o}^C_{t,i}$
$\{\mathbf{o}_{t,i}\}$
Multi-Query Attention (Core Attention)
$\{[\mathbf{q}^A_{t,i};\mathbf{q}^R_{t,i}]\}$
$\{[\mathbf{c}^{KV}_t;\mathbf{k}^R_t]\}$
concatenate
$\{\mathbf{q}^A_{t,i}\}$
$W^{UK}_i \mathbf{q}^C_{t,i}$
$\{\mathbf{q}^C_{t,i}\}$
$\{\mathbf{q}^R_{t,i}\}$
apply RoPE
$\mathbf{k}^R_t$
$\mathbf{c}^{KV}_t$
$\mathbf{c}^Q_t$
Input Hidden $\mathbf{h}_t$

(a) MHA mode of MLA. (b) MQA mode of MLA.

Figure 7 | Illustration of the MHA and MQA modes of MLA. For DeepSeek-V3.1-Terminus, the MHA mode is used for training and prefilling, while the MQA mode is used for decoding.

Figure 7 illustrates two aspects of MLA – the MHA and MQA modes – as well as the transformation between them.

B. Cold Start Template

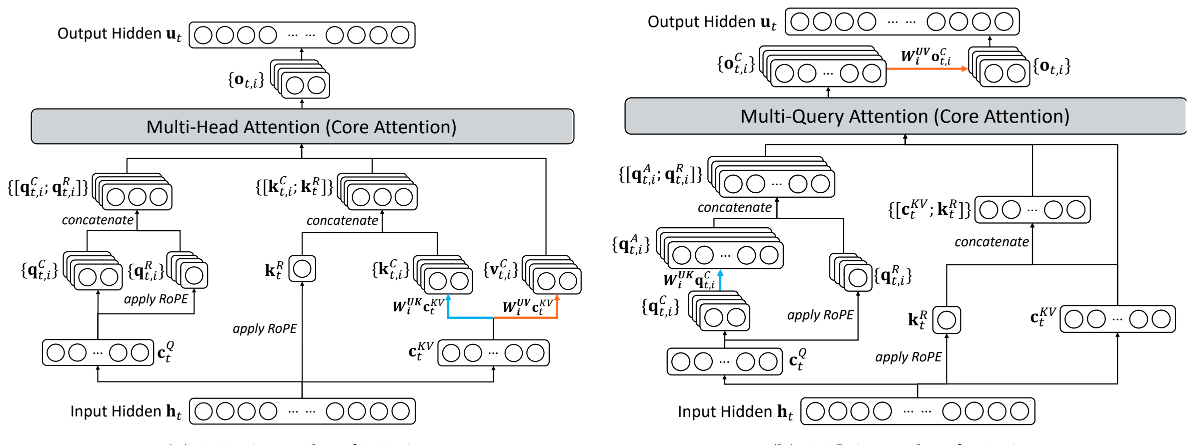
J. Yuan，H.高，D.戴，J.罗，L.赵，Z.Zhang，Z.Xie，Y.Wei，L.Wang，Z.Xiao，Y.Wang，C.Ruan，M.Zhang，W.Liang，和W.Zeng。原生稀疏注意力：硬件对齐且原生可训练的稀疏注意力。见 W. Che、J. Nabende、E. Shutova 和 M. T. Pile-hvar，编辑，计算语言学协会第 63 届年会论文集（第 1 卷：~~长论文），ACL 2025，第 23078-23097 页。计算语言学协会，2025。URL~~ ~~https://aclanthology.org/2025.acl-long~~.1126/。

智普-AI。 Glm-4.5：主体、推理和编码（弧）基础模型。 arXiv 预印本 arXiv:2508.06471, 2025。

周鹏，B. Leon，X. Ying，C. 张，Y. Shao，Q. Ye，D. Chong，Z. Jin，C. Xie，M. Cao，等。 Browsecomp-zh：中文大型语言模型的网页浏览能力基准测试。 arXiv 预印本 arXiv:2504.19314, 2025。

附录

MLA的A. MHA和MQA模式


Output Hidden $\mathbf{u}_t$
$\{\mathbf{o}_{t,i}\}$
Multi-Head Attention (Core Attention)
$\{[\mathbf{q}^C_{t,i}; \mathbf{q}^R_{t,i}]\}$
$\{[\mathbf{k}^C_{t,i}; \mathbf{k}^R_t]\}$
concatenate
concatenate
$\{\mathbf{q}^C_{t,i}\}$
$\{\mathbf{q}^R_{t,i}\}$
$\mathbf{k}^R_t$
$\{\mathbf{k}^C_{t,i}\}$
$\{\mathbf{v}^C_{t,i}\}$
apply RoPE
apply RoPE
$W^{UK}_i \mathbf{c}^{KV}_t$
$W^{UV}_i \mathbf{c}^{KV}_t$
$\mathbf{c}^Q_t$
$\mathbf{c}^{KV}_t$
Input Hidden $\mathbf{h}_t$
Output Hidden $\mathbf{u}_t$
$\{\mathbf{o}^C_{t,i}\}$
$W^{UV}_i \mathbf{o}^C_{t,i}$
$\{\mathbf{o}_{t,i}\}$
Multi-Query Attention (Core Attention)
$\{[\mathbf{q}^A_{t,i}; \mathbf{q}^R_{t,i}]\}$
concatenate
$\{[\mathbf{c}^{KV}_t; \mathbf{k}^R_t]\}$
$\{\mathbf{q}^A_{t,i}\}$
concatenate
$W^{UK}_i \mathbf{q}^C_{t,i}$
$\{\mathbf{q}^R_{t,i}\}$
$\{\mathbf{q}^C_{t,i}\}$
apply RoPE
$\mathbf{k}^R_t$
$\mathbf{c}^{KV}_t$
$\mathbf{c}^Q_t$
apply RoPE
Input Hidden $\mathbf{h}_t$

(a) MHA mode of MLA.　　(b) MQA mode of MLA.

图 7 | MLA 的 MHA 和 MQA 模式图示。对于DeepSeek-V3.1-Terminus，MHA模式用于训练和预填充，而MQA模式用于解码。

图 7 说明了 MLA 的两个方面——MHA 和 MQA 模式——以及它们之间的转换。

B. 冷启动模板

Table 6 | An example of the reasoning data system prompt. The system prompt requires the model to output the reasoning process in the tag <think></think>.

Reasoning 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. Please first reason before giving the final answer. The reasoning process enclosed within <think> </think>. The final answer is output after the </think> tag.
Prompt	Given a linked list, swap every two adjacent nodes and return its head ...
Reasoning Response	<think> ... </think> [FINAL ANSWER]

Table 7 | {TOOL-DESCRIPTIONS} and {TOOLCALL-FORMAT} will be replaced with the specific tools and our designed toolcall format.

Agent System Prompt	Use Python interpreter tool to execute Python code. The code will not be shown to the user. This tool should be used for internal reasoning, but not for code that is intended to be visible to the user (e.g. when creating plots, tables, or files). When you send a message containing Python code to python, it will be executed in a stateful Jupyter notebook environment. python will respond with the output of the execution or time out after 120.0 seconds. ## Tools You have access to the following tools: {TOOL-DESCRIPTIONS} Important: ALWAYS adhere to this exact format for tool use: {TOOLCALL-FORMAT}
Prompt	Given a linked list, swap every two adjacent nodes and return its head ...
Agent Response	[MULTI-TURN TOOLCALL] [FINAL ANSWER]

Table 8 | The model executes tool calls in thinking process.

Reasoning Required Agent System Prompt	You are a helpful assistant with access to a Python interpreter. - You may use the Python tool **multiple times** during your reasoning, a.k.a in <think></think>, with a maximum of 20 code executions. - Call the Python tool early in your reasoning to aid in solving the task. Continue reasoning and invoking tools as needed until you reach the final answer. Once you have the answer, stop reasoning and present your solution using Markdown and LaTeX. - Do NOT invoke any tools in your presented final solution steps. - To improve efficiency and accuracy, you should prefer code execution over language-based reasoning whenever possible. Keep your reasoning succinct; let the code do the heavy lifting. ## Tools You have access to the following tools: {TOOL-DESCRIPTIONS} Important: ALWAYS adhere to this exact format for tool use: {TOOLCALL-FORMAT}
Prompt	Given a linked list, swap every two adjacent nodes and return its head ...
Agent Response with Thinking	<think> [MULTI-TURN Thinking-Then-TOOLCALL] </think> [FINAL ANSWER]

Table 6 | An example of the reasoning data system prompt. The system prompt requires the model to output the reasoning process in the tag <think></think>.

Reasoning 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. Please first reason before giving the final answer. The reasoning process enclosed within <think> </think>. The final answer is output after the </think> tag.
Prompt	Given a linked list, swap every two adjacent nodes and return its head ...
Reasoning Response	<think> ... </think> [FINAL ANSWER]

Table 7 | {TOOL-DESCRIPTIONS} and {TOOLCALL-FORMAT} will be replaced with the specific tools and our designed toolcall format.

Agent System Prompt	Use Python interpreter tool to execute Python code. The code will not be shown to the user. This tool should be used for internal reasoning, but not for code that is intended to be visible to the user (e.g. when creating plots, tables, or files). When you send a message containing Python code to python, it will be executed in a stateful Jupyter notebook environment. python will respond with the output of the execution or time out after 120.0 seconds. ## Tools You have access to the following tools: {TOOL-DESCRIPTIONS} Important: ALWAYS adhere to this exact format for tool use: {TOOLCALL-FORMAT}
Prompt	Given a linked list, swap every two adjacent nodes and return its head ...
Agent Response	[MULTI-TURN TOOLCALL] [FINAL ANSWER]

Table 8 | The model executes tool calls in thinking process.

Reasoning Required Agent System Prompt	You are a helpful assistant with access to a Python interpreter. - You may use the Python tool **multiple times** during your reasoning, a.k.a in <think></think>, with a maximum of 20 code executions. - Call the Python tool early in your reasoning to aid in solving the task. Continue reasoning and invoking tools as needed until you reach the final answer. Once you have the answer, stop reasoning and present your solution using Markdown and LaTeX. - Do NOT invoke any tools in your presented final solution steps. - To improve efficiency and accuracy, you should prefer code execution over language-based reasoning whenever possible. Keep your reasoning succinct; let the code do the heavy lifting. ## Tools You have access to the following tools: {TOOL-DESCRIPTIONS} Important: ALWAYS adhere to this exact format for tool use: {TOOLCALL-FORMAT}
Prompt	Given a linked list, swap every two adjacent nodes and return its head ...
Agent Response with Thinking	<think> [MULTI-TURN Thinking-Then-TOOLCALL] </think> [FINAL ANSWER]

C. Non-thinking DeepSeek-V3.2 Agentic Evaluation

Table 9 | Comparison between DeepSeek-V3.2 non-thinking and thinking modes. The terminal bench scores are evaluated with the Claude Code framework in the table. Non-thinking score of Terminal Bench 2.0 with Terminus framework is 39.3.

	Benchmark (Metric)	**non-thinking**	**thinking**
Code Agent	Terminal Bench 2.0 (Acc)	37.1	46.4
	SWE Verified (Resolved)	72.1	73.1
	SWE Multilingual (Resolved)	68.9	70.2
ToolUse	τ^2-bench (Pass@1)	77.2	80.3
	MCP-Universe (Success Rate)	38.6	45.9
	MCP-Mark (Pass@1)	26.5	38.0
	Tool-Decathlon (Pass@1)	25.6	35.2

The performance of non-thinking mode is slightly worse than the thinking mode, but still competitive.

D. Evaluation Method of IOI, ICPC World Final, IMO, and CMO

For all competitions, the model's maximum generation length is set to 128k. No tools or internet access are used, and testing strictly adheres to the contest's time and attempt limits.

For the IOI evaluation, we designed our submission strategy in accordance with the official competition rules, which permit up to 50 submissions per problem and score each submission based on the maximum points achieved across all subtasks. Specifically, we first sampled 500 candidate solutions for each problem, then applied a multi-stage filtering pipeline. In the initial stage, we eliminated invalid submissions that failed to pass the provided sample test cases or exceeded the length constraints. Subsequently, we employed the DeepSeek-V32-Exp model to identify and remove samples in which the model explicitly indicated an inability or refusal to solve the problem. From the remaining valid candidates, we selected the 50 samples with the longest thinking traces for final submission.

For the ICPC evaluation, we adapted the same filtering methodology but with a smaller initial sample size. We generated 32 candidate solutions per problem and applied the identical filtering criteria to select submissions.

In the IMO and CMO tasks, we employ a generate-verify-refine loop. The model iteratively improves its solution until it achieves a perfect self-evaluation or hits the maximum revision cap, identical to the process in Shao et al. (2025).

C. Non-thinking DeepSeek-V3.2 Agentic Evaluation

Table 9 | Comparison between DeepSeek-V3.2 non-thinking and thinking modes. The terminal bench scores are evaluated with the Claude Code framework in the table. Non-thinking score of Terminal Bench 2.0 with Terminus framework is 39.3.

	Benchmark (Metric)	**non-thinking**	**thinking**
Code Agent	Terminal Bench 2.0 (Acc)	37.1	46.4
	SWE Verified (Resolved)	72.1	73.1
	SWE Multilingual (Resolved)	68.9	70.2
ToolUse	τ^2-bench (Pass@1)	77.2	80.3
	MCP-Universe (Success Rate)	38.6	45.9
	MCP-Mark (Pass@1)	26.5	38.0
	Tool-Decathlon (Pass@1)	25.6	35.2

The performance of non-thinking mode is slightly worse than the thinking mode, but still competitive.

D. Evaluation Method of IOI, ICPC World Final, IMO, and CMO

For all competitions, the model's maximum generation length is set to 128k. No tools or internet access are used, and testing strictly adheres to the contest's time and attempt limits.

For the IOI evaluation, we designed our submission strategy in accordance with the official competition rules, which permit up to 50 submissions per problem and score each submission based on the maximum points achieved across all subtasks. Specifically, we first sampled 500 candidate solutions for each problem, then applied a multi-stage filtering pipeline. In the initial stage, we eliminated invalid submissions that failed to pass the provided sample test cases or exceeded the length constraints. Subsequently, we employed the DeepSeek-V32-Exp model to identify and remove samples in which the model explicitly indicated an inability or refusal to solve the problem. From the remaining valid candidates, we selected the 50 samples with the longest thinking traces for final submission.

For the ICPC evaluation, we adapted the same filtering methodology but with a smaller initial sample size. We generated 32 candidate solutions per problem and applied the identical filtering criteria to select submissions.

In the IMO and CMO tasks, we employ a generate-verify-refine loop. The model iteratively improves its solution until it achieves a perfect self-evaluation or hits the maximum revision cap, identical to the process in Shao et al. (2025).

E. Author List

Research & Engineering: Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan*, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou*, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran*, Junguang Jiang, Junjie Qiu, Junlong Li*, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang*, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin*, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M.S. Di, M.Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S.H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li*, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y.Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong*, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z.F. Wu, Z.Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao

Data Annotation: Bei Feng, Hui Li, J.L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R.J. Chen, R.L. Jin, S.S. Li, Shuang Zhou, Tianyu Sun, X.Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y.X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang

Business & Compliance: Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T.Wang, W.L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, Zihua Qu

Authors are listed alphabetically by their first name. Names marked with * denote individuals who have departed from our team.

E. Author List

Research & Engineering: Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan*, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou*, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran*, Junguang Jiang, Junjie Qiu, Junlong Li*, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang*, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin*, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M.S. Di, M.Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S.H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li*, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y.Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong*, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z.F. Wu, Z.Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao

Data Annotation: Bei Feng, Hui Li, J.L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R.J. Chen, R.L. Jin, S.S. Li, Shuang Zhou, Tianyu Sun, X.Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y.X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang

Business & Compliance: Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T.Wang, W.L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, Zihua Qu

Authors are listed alphabetically by their first name. Names marked with * denote individuals who have departed from our team.