本文由 AINLP 公众号整理翻译，更多 LLM 资源请扫码关注!

KIMI K2: OPEN AGENTIC INTELLIGENCE

TECHNICAL REPORT OF KIMI K2

Kimi Team

ABSTRACT

We introduce Kimi K2, a Mixture-of-Experts (MoE) large language model with 32 billion activated parameters and 1 trillion total parameters. We propose the MuonClip optimizer, which improves upon Muon with a novel QK-clip technique to address training instability while enjoying the advanced token efficiency of Muon. Based on MuonClip, K2 was pre-trained on 15.5 trillion tokens with zero loss spike. During post-training, K2 undergoes a multi-stage post-training process, highlighted by a large-scale agentic data synthesis pipeline and a joint reinforcement learning (RL) stage, where the model improves its capabilities through interactions with real and synthetic environments.

Kimi K2 achieves state-of-the-art performance among open-source non-thinking models, with strengths in agentic capabilities. Notably, K2 obtains 66.1 on Tau2-Bench, 76.5 on ACEBench (En), 65.8 on SWE-Bench Verified, and 47.3 on SWE-Bench Multilingual — surpassing most open and closed-sourced baselines in non-thinking settings. It also exhibits strong capabilities in coding, mathematics, and reasoning tasks, with a score of 53.7 on LiveCodeBench v6, 49.5 on AIME 2025, 75.1 on GPQA-Diamond, and 27.1 on OJBench, all without extended thinking. These results position Kimi K2 as one of the most capable open-source large language models to date, particularly in software engineering and agentic tasks. We release our base and post-trained model checkpoints[1] to facilitate future research and applications of agentic intelligence.

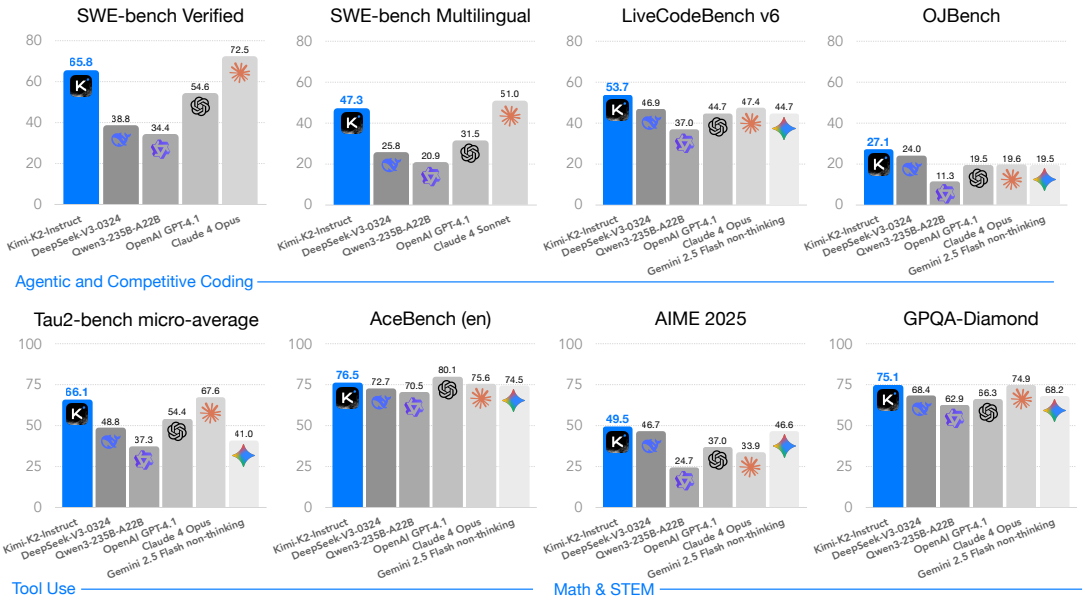


Figure 1: Kimi K2 main results.[2]

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

[2] All models evaluated above are non-thinking models. For SWE-bench Multilingual, we evaluated only Claude 4 Sonnet because the cost of Claude 4 Opus was prohibitive.

KIMI K2：开放智能体智能

KIMI K2 技术报告

Kimi 团队

摘要

我们推出 Kimi K2，这是一个拥有 320 亿激活参数、总计 1 万亿参数的混合专家（MoE）大语言模型。我们提出 MuonClip 优化器，在 Muon 基础上引入新颖的 QK-clip 技术，以解决训练不稳定问题，同时保留 Muon 先进的 token 效率。基于 MuonClip，K2 在 15.5 万亿 token 上进行了预训练，全程零损失尖峰。在后训练阶段，K2 经历多阶段后训练流程，核心包括大规模智能体数据合成管道和联合强化学习（RL）阶段，模型通过与真实及合成环境的交互持续提升能力。

Kimi K2 在开源非思维模型中实现了当前最佳性能，尤其在智能体能力方面表现突出。值得注意的是，K2 在 Tau2-Bench 上获得 66.1 分，在 ACEBench（英文）上获得 76.5 分，在 SWE-Bench Verified 上获得 65.8 分，在 SWE-Bench Multilingual 上获得 47.3 分——在非思维设置下均超越了大多数开源和闭源基线。它在编程、数学和推理任务上也展现出强大能力，在 LiveCodeBench v6 上得分 53.7，在 AIME 2025 上得分 49.5，在 GPQA-Diamond 上得分 75.1，在 OJBench 上得分 27.1，且均未使用扩展思维。这些结果使 Kimi K2 成为迄今为止能力最强的开源大语言模型之一，尤其在软件工程和智能体任务方面。我们发布了基础模型和训练后模型检查点[1]，以促进未来对智能体智能的研究与应用。

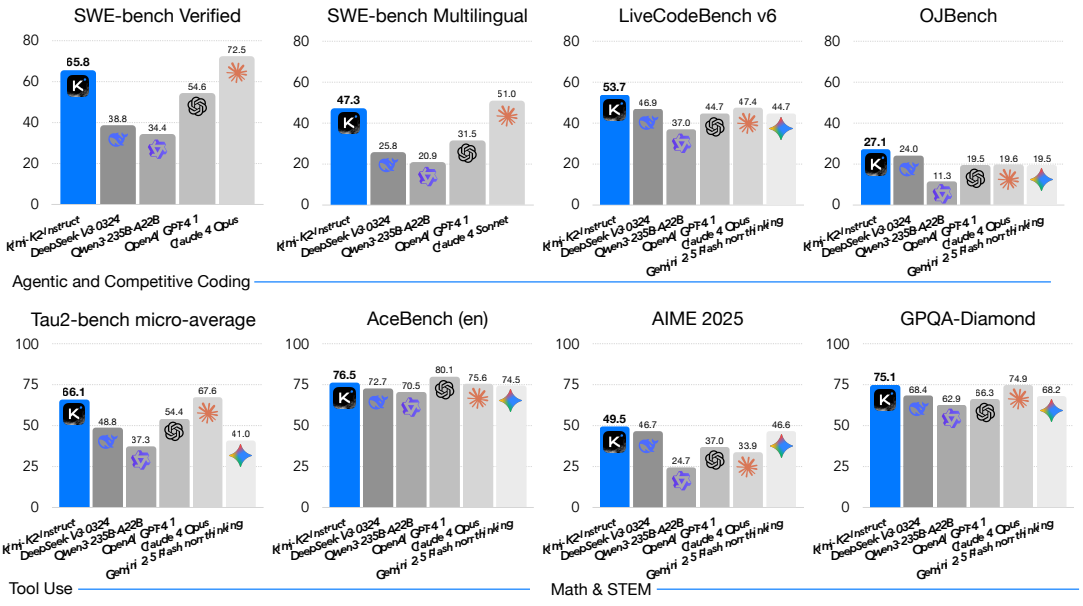


图1：Kimi K2 主要结果。[2]

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

[2] All models evaluated above are non-thinking models. For SWE-bench Multilingual, we evaluated only Claude 4 Sonnet because the cost of Claude 4 Opus was prohibitive.

1 Introduction

The development of Large Language Models (LLMs) is undergoing a profound paradigm shift towards *Agentic Intelligence* – the capabilities for models to autonomously perceive, plan, reason, and act within complex and dynamic environments. This transition marks a departure from static imitation learning towards models that actively learn through interactions, acquire new skills beyond their training distribution, and adapt behavior through experiences [63]. It is believed that this approach allows an AI agent to go beyond the limitation of static human-generated data, and acquire superhuman capabilities through its own exploration and exploitation. Agentic intelligence is thus rapidly emerging as a defining capability for the next generation of foundation models, with wide-ranging implications across tool use, software development, and real-world autonomy.

Achieving agentic intelligence introduces challenges in both pre-training and post-training. Pre-training must endow models with broad general-purpose priors under constraints of limited high-quality data, elevating token efficiency—learning signal per token—as a critical scaling coefficient. Post-training must transform those priors into actionable behaviors, yet agentic capabilities such as multi-step reasoning, long-term planning, and tool use are rare in natural data and costly to scale. Scalable synthesis of structured, high-quality agentic trajectories, combined with general reinforcement learning (RL) techniques that incorporate preferences and self-critique, are essential to bridge this gap.

In this work, we introduce Kimi K2, a 1.04 trillion-parameter Mixture-of-Experts (MoE) LLM with 32 billion activated parameters, purposefully designed to address the core challenges and push the boundaries of agentic capability. Our contributions span both the pre-training and post-training frontiers:

- We present **MuonClip**, a novel optimizer that integrates the token-efficient Muon algorithm with a stability-enhancing mechanism called QK-Clip. Using MuonClip, we successfully pre-trained Kimi K2 on 15.5 trillion tokens without a single loss spike.
- We introduce **a large-scale agentic data synthesis pipeline** that systematically generates tool-use demonstrations via simulated and real-world environments. This system constructs diverse tools, agents, tasks, and trajectories to create high-fidelity, verifiably correct agentic interactions at scale.
- We design **a general reinforcement learning framework** that combines verifiable rewards (RLVR) with a self-critique rubric reward mechanism. The model learns not only from externally defined tasks but also from evaluating its own outputs, extending alignment from static into open-ended domains.

Kimi K2 demonstrates strong performance across a broad spectrum of agentic and frontier benchmarks. It achieves scores of 66.1 on Tau2-bench, 76.5 on ACEBench (en), 65.8 on SWE-bench Verified, and 47.3 on SWE-bench Multilingual, outperforming most open- and closed-weight baselines under non-thinking evaluation settings, closing the gap with Claude 4 Opus and Sonnet. In coding, mathematics, and broader STEM domains, Kimi K2 achieves 53.7 on LiveCodeBench v6, 27.1 on OJBench, 49.5 on AIME 2025, and 75.1 on GPQA-Diamond, further highlighting its capabilities in general tasks. On the LMSYS Arena leaderboard (July 17, 2025)[3], Kimi K2 ranks as the top 1 open-source model and 5th overall based on over 3,000 user votes.

To spur further progress in Agentic Intelligence, we are open-sourcing our base and post-trained checkpoints, enabling the community to explore, refine, and deploy agentic intelligence at scale.

2 Pre-training

The base model of Kimi K2 is a trillion-parameter mixture-of-experts (MoE) transformer [72] model, pre-trained on 15.5 trillion high-quality tokens. Given the increasingly limited availability of high-quality human data, we posit that token efficiency is emerging as a critical coefficient in the scaling of large language models. To address this, we introduce a suite of pre-training techniques explicitly designed for maximizing token efficiency. Specifically, we employ the token-efficient Muon optimizer [33, 46] and mitigate its training instabilities through the introduction of QK-Clip. Additionally, we incorporate synthetic data generation to further squeeze the intelligence out of available high-quality tokens. The model architecture follows an ultra-sparse MoE with multi-head latent attention (MLA) similar to DeepSeek-V3 [10] , derived from empirical scaling law analysis. The underlying infrastructure is built to optimize both training efficiency and research efficiency.

[3] https://lmarena.ai/leaderboard/text

1 引言

大型语言模型（LLMs）的发展正经历一场深刻的范式转变，迈向*Agentic Intelligence*——模型在复杂动态环境中自主感知、规划、推理与行动的能力。这一转变标志着从静态模仿学习向通过交互主动学习的模型过渡，使其能够超越训练分布获取新技能，并通过经验调整行为[63]。人们相信，这种方法使AI智能体突破静态人类生成数据的局限，通过自身探索与利用获得超人能力。因此，智能体智能正迅速成为下一代基础模型的核心能力，对工具使用、软件开发和现实世界自主性产生广泛影响。

实现代理智能在预训练和后训练阶段都带来了挑战。预训练必须在高质量数据有限的约束下，赋予模型广泛的通用先验，从而提升 token 效率——即每个 token 的学习信号——使其成为关键的扩展系数。后训练则必须将这些先验转化为可执行的行为，然而诸如多步推理、长期规划和工具使用等代理能力在自然数据中极为稀少，且扩展成本高昂。可扩展地合成结构化、高质量的代理轨迹，并结合能够纳入偏好与自我批判的通用强化学习（RL）技术，是弥合这一差距的关键。

在这项工作中，我们推出 Kimi K2，一个拥有 1.04 万亿参数的混合专家（MoE）大语言模型，其中 320 亿参数被激活，专门设计用于解决核心挑战并拓展智能体能力的边界。我们的贡献涵盖预训练与后训练两大前沿：

- 我们提出 MuonClip，这是一种将 token 高效的 Muon 算法与名为 QK-Clip 的稳定性增强机制相结合的新型优化器。借助 MuonClip，我们成功地在 15.5 万亿个 token 上预训练了 Kimi K2，期间未出现任何损失尖峰。
- 我们引入了一个大规模智能体数据合成流水线，通过模拟和真实环境系统地生成工具使用演示。该系统构建多样化的工具、智能体、任务和轨迹，以大规模创建高保真、可验证正确的智能体交互。
- 我们设计了一个通用强化学习框架，将可验证奖励（RLVR）与自评量表奖励机制相结合。该模型不仅从外部定义的任务中学习，还通过评估自身输出来学习，将对齐从静态领域扩展到开放式领域。

Kimi K2 在广泛的智能体与前沿基准测试中展现出强劲表现：在 Tau2-bench 上得分 66.1，ACEBench（en）76.5，SWE-bench Verified 65.8，SWE-bench Multilingual 47.3，在非思考评估设置下超越大多数开源与闭源基线，缩小了与 Claude 4 Opus 和 Sonnet 的差距。在编程、数学及更广泛的 STEM 领域，Kimi K2 在 LiveCode Bench v6 取得 53.7，OJBench 27.1，AIME 2025 49.5，GPQA-Diamond 75.1，进一步凸显其在通用任务上的能力。在 LMSYS Arena 排行榜（2025 年 7 月 17 日）[3] 上，Kimi K2 以超过 3,000 张用户投票位列开源模型第 1、总体第 5。

为了进一步推动智能体智能的发展，我们开源了基础和后训练检查点，使社区能够大规模地探索、优化和部署智能体智能。

2 预训练

Kimi K2 的基础模型是一个万亿参数的混合专家（MoE）Transformer [72] 模型，在 15.5 万亿高质量 token 上进行了预训练。鉴于高质量人类数据日益稀缺，我们认为 token 效率正成为大语言模型规模扩展中的关键系数。为此，我们引入了一套专为最大化 token 效率而设计的预训练技术。具体而言，我们采用了 token 高效的 Muon 优化器 [33, 46]，并通过引入 QK-Clip 来缓解其训练不稳定性。此外，我们还引入合成数据生成，以进一步从可用的高质量 token 中“榨取”智能。模型架构采用与 DeepSeek-V3 [10] 类似的超稀疏 MoE 与多头潜在注意力（MLA），源自经验性缩放律分析。底层基础设施旨在同时优化训练效率与研究效率。

[3] https://lmarena.ai/leaderboard/text

2.1 MuonClip: Stable Training with Weight Clipping

We train Kimi K2 using the token-efficient Muon optimizer [33], incorporating weight decay and consistent update RMS scaling [46]. Experiments in our previous work Moonlight [46] show that, under the same compute budget and model size — and therefore the same amount of training data — Muon substantially outperforms AdamW [36, 48], making it an effective choice for improving token efficiency in large language model training.

Training instability when scaling Muon Despite its efficiency, scaling up Muon training reveals a challenge: training instability due to exploding attention logits, an issue that occurs more frequently with Muon but less with AdamW in our experiments. Existing mitigation strategies are insufficient. For instance, logit soft-cap [69] directly clips the attention logits, but the dot products between queries and keys can still grow excessively before capping is applied. On the other hand, Query-Key Normalization (QK-Norm) [11, 81] is not applicable to multi-head latent attention (MLA), because its Key matrices are not fully materialized during inference.

Taming Muon with QK-Clip To address this issue, we propose a novel weight-clipping mechanism *QK-Clip* to explicitly constrain attention logits. QK-Clip works by rescaling the query and key projection weights post-update to bound the growth of attention logits.

Let the input representation of a transformer layer be $\mathbf{X}$. For each attention head h, its query, key, and value projections are computed as

$$\mathbf{Q}^h = \mathbf{X}\mathbf{W}_q^h, \quad \mathbf{K}^h = \mathbf{X}\mathbf{W}_k^h, \quad \mathbf{V}^h = \mathbf{X}\mathbf{W}_v^h.$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ are model parameters. The attention output is:

$$\mathbf{O}^h = \mathrm{softmax}\left(\frac{1}{\sqrt{d}}\mathbf{Q}^h\mathbf{K}^{h\top}\right)\mathbf{V}^h.$$

We define the max logit, a per-head scalar, as the maximum input to softmax in this batch B:

$$S_{\max}^h = \frac{1}{\sqrt{d}} \max_{\mathbf{X}\in B} \max_{i,j} \mathbf{Q}_i^h \mathbf{K}_j^{h\top}$$

where i, j are indices of different tokens in a training sample $\mathbf{X}$.

The core idea of QK-Clip is to rescale $\mathbf{W}_k, \mathbf{W}_q$ whenever $S_{\max}^h$ exceeds a target threshold τ. Importantly, this operation does not alter the forward/backward computation in the current step — we merely use the max logit as a guiding signal to determine the strength to control the weight growth.

A naïve implementation clips all heads at the same time:

$$\mathbf{W}_q^h \leftarrow \gamma^\alpha \mathbf{W}_q^h \qquad \mathbf{W}_k^h \leftarrow \gamma^{1-\alpha}\mathbf{W}_k^h$$

where $\gamma = \min(1, \tau/S_{\max})$ with $S_{\max} = \max_h S_{\max}^h$, and α is a balancing parameter typically set to 0.5, applying equal scaling to queries and keys.

However, we observe that in practice, only a small subset of heads exhibit exploding logits. In order to minimize our intervention on model training, we determine a per-head scaling factor $\gamma_h = \min(1, \tau/S_{\max}^h)$, and opt to apply per-head QK-Clip. Such clipping is straightforward for regular multi-head attention (MHA). For MLA, we apply clipping only on unshared attention head components:

- $\mathbf{q}^C$ and $\mathbf{k}^C$ (head-specific components): each scaled by $\sqrt{\gamma_h}$
- $\mathbf{q}^R$ (head-specific rotary): scaled by γ_h,
- $\mathbf{k}^R$ (shared rotary): left untouched to avoid effect across heads.

MuonClip: The New Optimizer We integrate Muon with weight decay, consistent RMS matching, and QK-Clip into a single optimizer, which we refer to as **MuonClip** (see Algorithm 1).

We demonstrate the effectiveness of MuonClip from several scaling experiments. First, we train a mid-scale 9B activated and 53B total parameters Mixture-of-Experts (MoE) model using the vanilla Muon. As shown in Figure 2 (Left), we observe that the maximum attention logits quickly exceed a magnitude of 1000, showing that attention logits explosion is already evident in Muon training to this scale. Max logits at this level usually result in instability during training, including significant loss spikes and occasional divergence.

2.1 MuonClip：通过权重裁剪实现稳定训练

我们使用 token 高效的 Muon 优化器 [33] 训练 Kimi K2，并结合权重衰减和一致的更新 RMS 缩放 [46]。在我们先前的工作 Moonlight [46] 中的实验表明，在相同的计算预算和模型规模下——因此训练数据量也相同——Muon 显著优于 AdamW [36, 48]，这使其成为提升大语言模型训练 token 效率的有效选择。

扩展 Muon 时的训练不稳定性 尽管 Muon 高效，但扩大其训练规模会暴露一个挑战：由于注意力 logit 爆炸导致的训练不稳定，这一问题在我们的实验中 Muon 出现得更频繁，而 AdamW 则较少。现有的缓解策略并不充分。例如，logit soft-cap [69] 直接裁剪注意力 logit，但在裁剪生效前，查询与键的点积仍可能过度增长。另一方面，Query-Key Normalization（QK-Norm）[11, 81] 不适用于多头潜在注意力（MLA），因为其 Key 矩阵在推理期间并未完全实例化。

使用 QK-Clip 驯服 Muon 为了解决这一问题，我们提出了一种新颖的权重裁剪机制 *QK-Clip*，以显式约束注意力 logits。QK-Clip 通过在更新后对查询和键的投影权重进行重缩放，从而限制注意力 logits 的增长。

设一个 Transformer 层的输入表示为 $\mathbf{X}$。对于每个注意力头 h，其查询、键和值的投影计算如下

$$\mathbf{Q}^h = \mathbf{X}\mathbf{W}_q^h, \quad \mathbf{K}^h = \mathbf{X}\mathbf{W}_k^h, \quad \mathbf{V}^h = \mathbf{X}\mathbf{W}_v^h.$$

其中 $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ 是模型参数。注意力输出为：

$$\mathbf{O}^h = \mathrm{softmax}\left(\frac{1}{\sqrt{d}}\mathbf{Q}^h\mathbf{K}^{h\top}\right)\mathbf{V}^h.$$

我们将最大 logit（每个注意力头的一个标量）定义为该批次 B 中 softmax 的最大输入：

$$S_{\max}^h = \frac{1}{\sqrt{d}} \max_{\mathbf{X}\in B} \max_{i,j} \mathbf{Q}_i^h \mathbf{K}_j^{h\top}$$

其中 i, j 是训练样本 $\mathbf{X}$ 中不同标记的索引。

QK-Clip 的核心思想是：当 $S_{\max}^h$ 超过目标阈值 τ 时，对 $\mathbf{W}_k, \mathbf{W}_q$ 进行重新缩放。重要的是，这一操作不会改变当前步骤的前向/反向计算——我们只是将最大 logit 作为引导信号，以确定控制权重增长的强度。

一种朴素的实现会同时裁剪所有头：

$$\mathbf{W}_q^h \leftarrow \gamma^{\alpha}\mathbf{W}_q^h \qquad \mathbf{W}_k^h \leftarrow \gamma^{1-\alpha}\mathbf{W}_k^h$$

其中 $\gamma = \min(1, \tau/S_{\max})$，且 $S_{\max} = \max_h S_{\max}^h$，而 α 是一个平衡参数，通常设为 0.5，对查询和键施加相同的缩放。

然而，我们观察到，在实践中只有一小部分注意力头会出现 logits 爆炸。为了将对模型训练的干预降至最低，我们为每个头确定一个缩放因子 $\gamma_h = \min(1, \tau/S_{\max}^h)$，并选择对每个头应用 QK-Clip。对于常规多头注意力（MHA），这种裁剪非常直接。对于 MLA，我们仅对未共享的注意力头分量进行裁剪：

- q^C 和 k^C (头特定组件)：每个都按 $\sqrt{\gamma_h}$ 缩放
- q^R (头特定的旋转)：按γ_h缩放，
- k^R (共享旋转)：保持不变，以避免跨头产生影响。

MuonClip：新的优化器 我们将 Muon 与权重衰减、一致的 RMS 匹配和 QK-Clip 整合为一个优化器，我们称之为 MuonClip（见算法 1）。

我们通过一系列扩展实验展示了 MuonClip 的有效性。首先，我们使用 vanilla Muon 训练了一个中等规模的 9B 激活、53B 总参数的混合专家（MoE）模型。如图 2（左）所示，我们观察到最大注意力 logits 迅速超过 1000 的量级，表明在 Muon 训练到这一规模时，注意力 logits 爆炸已经十分明显。达到这一水平的最大 logits 通常会导致训练不稳定，包括显著的损失尖峰和偶发的发散。

Algorithm 1 MuonClip Optimizer

for each training step t **do**
  // 1. Muon optimizer step
  for each weight $\mathbf{W} \in \mathbb{R}^{n \times m}$ **do**
    $\mathbf{M}_t = \mu \mathbf{M}_{t-1} + \mathbf{G}_t$ ▷ $\mathbf{M}_0 = \mathbf{0}$, $\mathbf{G}_t$ is the grad of $\mathbf{W}_t$, μ is momentum
    $\mathbf{O}_t = \text{Newton-Schulz}(\mathbf{M}_t) \cdot \sqrt{\max(n, m)} \cdot 0.2$ ▷ Match Adam RMS
    $\mathbf{W}_t = \mathbf{W}_{t-1} - \eta\big(\mathbf{O}_t + \lambda \mathbf{W}_{t-1}\big)$ ▷ learning rate η, weight decay λ
  end for
  // 2. QK-Clip
  for each attention head h in every attention layer of the model **do**
    Obtain $S^h_{\max}$ already computed during forward
    if $S^h_{\max} > \tau$ **then**
      $\gamma \leftarrow \tau / S^h_{\max}$
      $\mathbf{W}^h_{qc} \leftarrow \mathbf{W}^h_{qc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{kc} \leftarrow \mathbf{W}^h_{kc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{qr} \leftarrow \mathbf{W}^h_{qr} \cdot \gamma$
    end if
  end for
end for

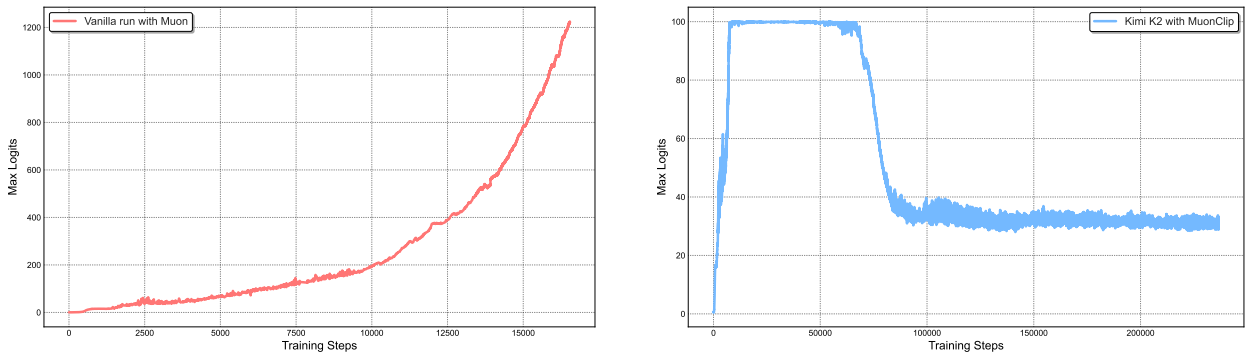


Figure 2: Left: During a mid-scale training run, attention logits rapidly exceed 1000, which could lead to potential numerical instabilities and even training divergence. Right: Maximum logits for Kimi K2 with MuonClip and τ = 100 over the entire training run. The max logits rapidly increase to the capped value of 100, and only decay to a stable range after approximately 30% of the training steps, demonstrating the effective regulation effect of QK-Clip.

Next, we demonstrate that QK-Clip does not degrade model performance and confirm that the MuonClip optimizer preserves the optimization characteristics of Muon without adversely affecting the loss trajectory. A detailed discussion of the experiment designs and findings is provided in the Appendix D.

Finally, we train Kimi K2, a large-scale MoE model, using MuonClip with $\tau = 100$ and monitor the maximum attention logits throughout the training run (Figure 2 (Right)). Initially, the logits are capped at 100 due to QK-Clip. Over the course of training, the maximum logits gradually decay to a typical operating range without requiring any adjustment to τ. Importantly, the training loss remains smooth and stable, with no observable spikes, as shown in Figure 3, validating that MuonClip provides robust and scalable control over attention dynamics in large-scale language model training.

2.2 Pre-training Data: Improving Token Utility with Rephrasing

Token efficiency in pre-training refers to how much performance improvement is achieved for each token consumed during training. Increasing token utility—the effective learning signal each token contributes—enhances the per-token impact on model updates, thereby directly improving token efficiency. This is particularly important when the supply of high-quality tokens is limited and must be maximally leveraged. A naive approach to increasing token utility is through repeated exposure to the same tokens, which can lead to overfitting and reduced generalization.

Algorithm 1 MuonClip Optimizer

for each training step t **do**
  // 1. Muon optimizer step
  for each weight $\mathbf{W} \in \mathbb{R}^{n\times m}$ **do**
    $\mathbf{M}_t = \mu\mathbf{M}_{t-1} + \mathbf{G}_t$ ▷ $\mathbf{M}_0 = \mathbf{0}$, $\mathbf{G}_t$ is the grad of $\mathbf{W}_t$, μ is momentum
    $\mathbf{O}_t = \text{Newton-Schulz}(\mathbf{M}_t) \cdot \sqrt{\max(n,m)} \cdot 0.2$ ▷ Match Adam RMS
    $\mathbf{W}_t = \mathbf{W}_{t-1} - \eta\big(\mathbf{O}_t + \lambda\mathbf{W}_{t-1}\big)$ ▷ learning rate η, weight decay λ
  end for
  // 2. QK-Clip
  for each attention head h in every attention layer of the model **do**
    Obtain $S^h_{\max}$ already computed during forward
    if $S^h_{\max} > \tau$ **then**
      $\gamma \leftarrow \tau / S^h_{\max}$
      $\mathbf{W}^h_{qc} \leftarrow \mathbf{W}^h_{qc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{kc} \leftarrow \mathbf{W}^h_{kc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{qr} \leftarrow \mathbf{W}^h_{qr} \cdot \gamma$
    end if
  end for
end for

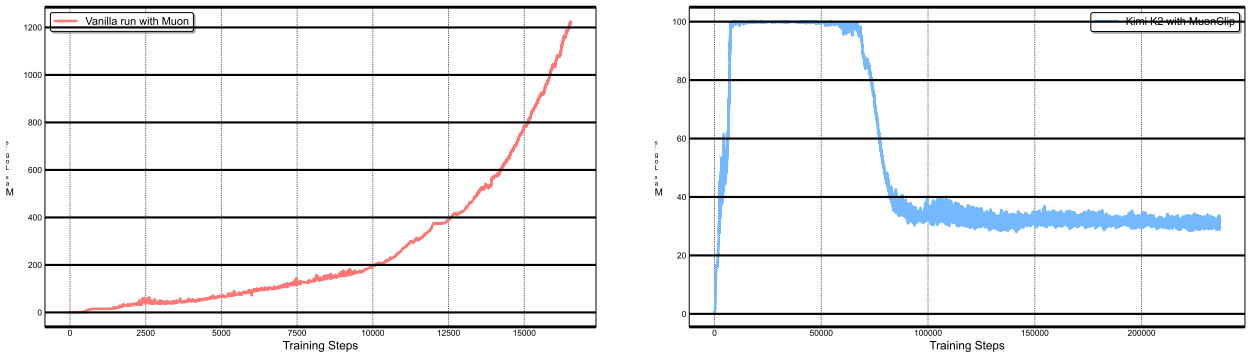


图2：左：在一次中等规模的训练过程中，注意力logits迅速超过1000，可能导致数值不稳定甚至训练发散。右：Kimi K2在整个训练过程中使用MuonClip和τ = 100时的最大logits。最大logits迅速上升至被限制的值100，并在大约30%的训练步数后才衰减到稳定区间，展示了QK-Clip的有效调节作用。

接下来，我们证明 QK-Clip 不会降低模型性能，并确认 MuonClip 优化器在不影响损失轨迹的前提下保留了 Muon 的优化特性。实验设计与发现的详细讨论见附录 D。

最后，我们使用 MuonClip 以 $\tau = 100$ 训练大规模 MoE 模型 Kimi K2，并在整个训练过程中监控最大注意力 logits（图 2 右）。初始时，由于 QK-Clip，logits 被限制在 100。随着训练进行，最大 logits 逐渐衰减到典型工作范围，而无需对 τ 做任何调整。重要的是，如图 3 所示，训练损失保持平滑稳定，未观察到任何尖峰，验证了 MuonClip 在大规模语言模型训练中为注意力动态提供了稳健且可扩展的控制。

2.2 预训练数据：通过改写提升标记效用

预训练中的Token效率指的是训练过程中每消耗一个Token所获得的性能提升。提高Token利用率——即每个Token贡献的有效学习信号——可以增强每个Token对模型更新的影响，从而直接提升Token效率。当高质量Token的供应有限且必须被最大化利用时，这一点尤为重要。一种简单提高Token利用率的方法是对相同Token进行重复曝光，但这可能导致过拟合并降低泛化能力。

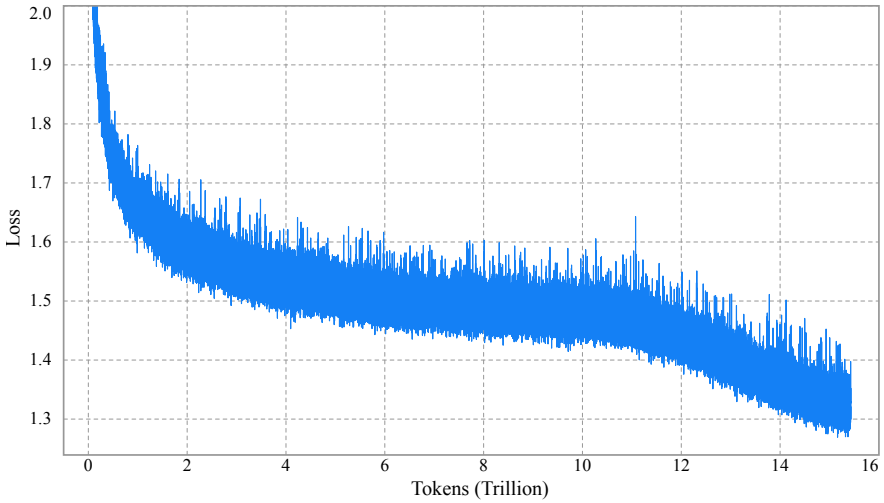


Figure 3: Per-step training loss curve of Kimi K2, **without smoothing or sub-sampling**. It shows no spikes throughout the entire training process. Note that we omit the very beginning of training for clarity.

A key advancement in the pre-training data of Kimi K2 over Kimi K1.5 is the introduction of a synthetic data generation strategy to increase token utility. Specifically, a carefully designed rephrasing pipeline is employed to amplify the volume of high-quality tokens without inducing significant overfitting. In this report, we describe two domain-specialized rephrasing techniques—targeted respectively at the Knowledge and Mathematics domains—that enable this controlled data augmentation.

Knowledge Data Rephrasing Pre-training on natural, knowledge-intensive text presents a trade-off: a single epoch is insufficient for comprehensive knowledge absorption, while multi-epoch repetition yields diminishing returns and increases the risk of overfitting. To improve the token utility of high-quality knowledge tokens, we propose a synthetic rephrasing framework composed of the following key components:

- **Style- and perspective-diverse prompting:** To enhance linguistic diversity while maintaining factual integrity, we apply a range of carefully engineered prompts. These prompts guide a large language model to generate faithful rephrasings of the original texts in varied styles and from different perspectives.
- **Chunk-wise autoregressive generation:** To preserve global coherence and avoid information loss in long documents, we adopt a chunk-based autoregressive rewriting strategy. Texts are divided into segments, rephrased individually, and then stitched back together to form complete passages. This method mitigates implicit output length limitations that typically exist with LLMs. An overview of this pipeline is presented in Figure 4.
- **Fidelity verification:** To ensure consistency between original and rewritten content, we perform fidelity checks that compare the semantic alignment of each rephrased passage with its source. This serves as an initial quality control step prior to training.

We compare data rephrasing with multi-epoch repetition by testing their corresponding accuracy on SimpleQA. We experiment with an early checkpoint of K2 and evaluate three training strategies: (1) repeating the original dataset for 10 epochs, (2) rephrasing the data once and repeating it for 10 epochs, and (3) rephrasing the data 10 times with a single training pass. As shown in Table 1, the accuracy consistently improves across these strategies, demonstrating the efficacy of our rephrasing-based augmentation. We extended this method to other large-scale knowledge corpora and observed similarly encouraging results, and each corpora is rephrased at most twice.

Table 1: SimpleQA Accuracy under three rephrasing-epoch configurations

# Rephrasings	# Epochs	SimpleQA Accuracy
0 (raw wiki-text)	10	23.76
1	10	27.39
10	1	28.94

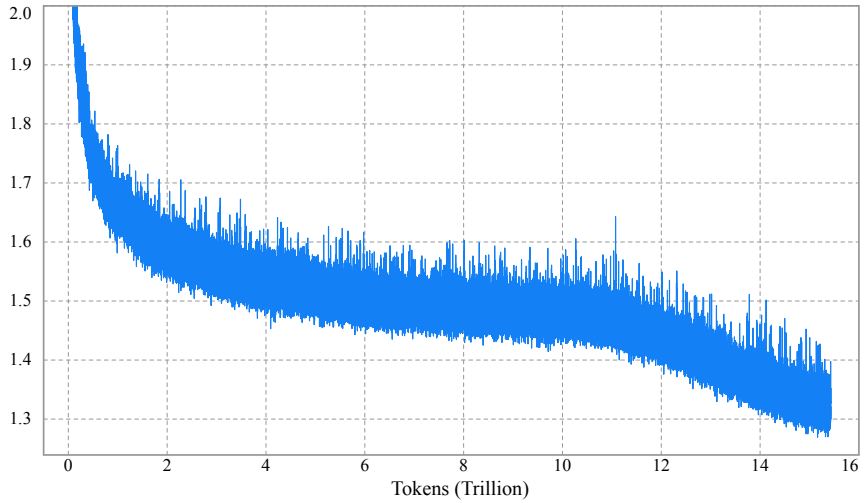


图3：Kimi K2 的每步训练损失曲线，未做平滑或子采样。整个训练过程中未出现任何尖峰。为清晰起见，我们省略了训练的最开始部分。

Kimi K2 在预训练数据上相较于 Kimi K1.5 的一项关键进展，是引入了合成数据生成策略以提升 token 的利用效率。具体而言，我们采用了一套精心设计的改写流水线，在不引发显著过拟合的前提下，放大高质量 token 的数量。在本报告中，我们描述了两种面向特定领域的改写技术——分别针对知识领域和数学领域——以实现这种受控的数据增强。

知识数据改写预训练：在自然、知识密集型文本上进行预训练面临一个权衡：单轮训练不足以全面吸收知识，而多轮重复训练则收益递减并增加过拟合风险。为了提高高质量知识 token 的 token 效用，我们提出一个由以下关键组件构成的合成改写框架：

- 风格与视角多样化的提示：为了在保持事实完整性的同时增强语言多样性，我们采用了一系列精心设计的提示。这些提示引导大型语言模型以不同风格和视角对原文进行忠实改写。
- 分块自回归生成：为了保持长文档的全局连贯性并避免信息丢失，我们采用基于分块的自回归重写策略。文本被划分为若干片段，分别进行改写，然后再拼接成完整的段落。该方法缓解了通常存在于大语言模型中的隐式输出长度限制。该流程的概览如图4所示。
- 保真度验证：为确保原始内容与重写内容的一致性，我们执行保真度检查，比较每个改写段落与其来源在语义上的一致性。这是在训练之前进行的初步质量控制步骤。

我们在 SimpleQA 上比较数据改写与多轮重复的效果，测试它们对应的准确率。我们使用 K2 的一个早期检查点进行实验，并评估三种训练策略：(1) 原始数据集重复训练 10 轮，(2) 数据改写一次后重复训练 10 轮，(3) 数据改写 10 次且仅训练一轮。如表 1 所示，这些策略的准确率持续提升，证明了我们基于改写的数据增强方法的有效性。我们将该方法扩展到其他大规模知识语料库，并观察到同样令人鼓舞的结果，每个语料库最多改写两次。

表1：三种重述轮次配置下的SimpleQA准确率

# Rephrasings	# Epochs	SimpleQA Accuracy
0 (raw wiki-text)	10	23.76
1	10	27.39
10	1	28.94

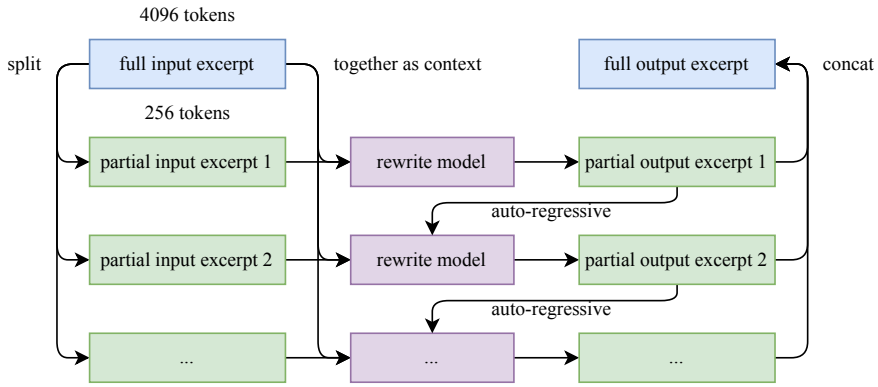


Figure 4: Auto-regressive chunk-wise rephrasing pipeline for long input excerpts. The input is split into smaller chunks with preserved context, rewritten sequentially, and then concatenated into a full rewritten passage.

Mathematics Data Rephrasing To enhance mathematical reasoning capabilities, we rewrite high-quality mathematical documents into a "learning-note" style, following the methodology introduced in SwallowMath [15]. In addition, we increased data diversity by translating high-quality mathematical materials from other languages into English.

Although initial experiments with rephrased subsets of our datasets show promising results, the use of synthetic data as a strategy for continued scaling remains an active area of investigation. Key challenges include generalizing the approach to diverse source domains without compromising factual accuracy, minimizing hallucinations and unintended toxicity, and ensuring scalability to large-scale datasets.

Pre-training Data Overall The Kimi K2 pre-training corpus comprises 15.5 trillion tokens of curated, high-quality data spanning four primary domains: Web Text, Code, Mathematics, and Knowledge. Most data processing pipelines follow the methodologies outlined in Kimi K1.5 [35]. For each domain, we performed rigorous correctness and quality validation and designed targeted data experiments to ensure the curated dataset achieved both high diversity and effectiveness.

2.3 Model Architecture

Kimi K2 is a 1.04 trillion-parameter Mixture-of-Experts (MoE) transformer model with 32 billion activated parameters. The architecture follows a similar design to DeepSeek-V3 [10] , employing Multi-head Latent Attention (MLA) [44] as the attention mechanism, with a model hidden dimension of 7168 and an MoE expert hidden dimension of 2048. Our scaling law analysis reveals that continued increases in sparsity yield substantial performance improvements, which motivated us to increase the number of experts to 384, compared to 256 in DeepSeek-V3. To reduce computational overhead during inference, we cut the number of attention heads to 64, as opposed to 128 in DeepSeek-V3. Table 2 presents a detailed comparison of architectural parameters between Kimi K2 and DeepSeek-V3.

Table 2: Architectural comparison between Kimi K2 and DeepSeek-V3

	DeepSeek-V3	**Kimi K2**	Δ
#Layers	61	61	=
Total Parameters	671B	1.04T	$\uparrow$ 54%
Activated Parameters	37B	32.6B	$\downarrow$ 13%
Experts (total)	256	384	$\uparrow$ 50%
Experts Active per Token	8	8	=
Shared Experts	1	1	=
Attention Heads	128	64	$\downarrow$ 50%
Number of Dense Layers	3	1	$\downarrow$ 67%
Expert Grouping	Yes	No	-

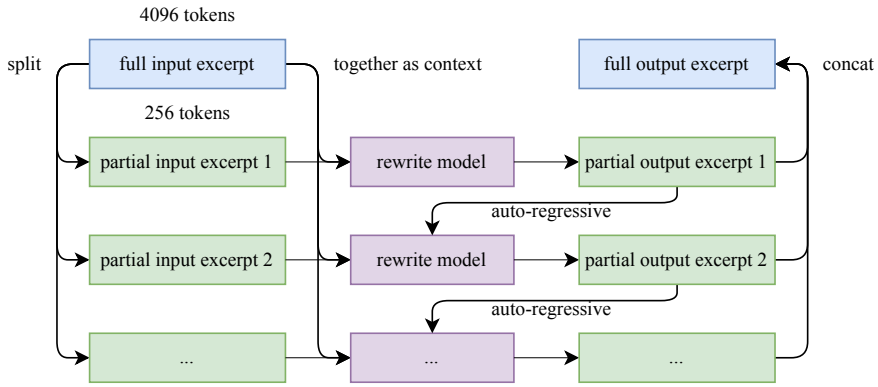


图4：用于长输入摘录的自回归分块改写流程。输入被拆分为保留上下文的小块，依次重写，然后拼接成完整的改写段落。

数学数据改写 为了提升数学推理能力，我们按照 SwallowMath [15] 提出的方法，将高质量数学文档改写成“学习笔记”风格。此外，我们还通过将其他语言的高质量数学资料翻译成英文来增加数据多样性。

尽管对我们数据集的改写子集进行的初步实验显示出可喜的结果，但将合成数据作为持续扩展策略的使用仍是一个活跃的研究领域。关键挑战包括将该方法推广到多样化的源领域而不损害事实准确性、最小化幻觉和意外的毒性，以及确保对大规模数据集的扩展性。

预训练数据概览 Kimi K2 预训练语料库共包含 15.5 万亿个经过精心筛选的高质量 token，覆盖四大主要领域：网页文本、代码、数学和知识。大多数数据处理流程沿用 Kimi K1.5 [35] 中描述的方法。针对每个领域，我们都进行了严格的正确性与质量验证，并设计了针对性的数据实验，以确保所构建的数据集兼具高多样性与高效能。

2.3 模型架构

Kimi K2 是一个拥有 1.04 万亿参数的混合专家（MoE）Transformer 模型，其中激活参数为 320 亿。其架构设计类似于 DeepSeek-V3 [10]，采用多头潜在注意力（MLA）[44] 作为注意力机制，模型隐藏维度为 7168，MoE 专家隐藏维度为 2048。我们的扩展定律分析表明，继续增加稀疏性可带来显著的性能提升，这促使我们将专家数量从 DeepSeek-V3 的 256 增加到 384。为降低推理时的计算开销，我们将注意力头数从 DeepSeek-V3 的 128 减少到 64。表 2 详细对比了 Kimi K2 与 DeepSeek-V3 的架构参数。

表2：Kimi K2 与 DeepSeek-V3 的架构对比

	DeepSeek-V3	**Kimi K2**	Δ
#Layers	61	61	=
Total Parameters	671B	1.04T	↑ 54%
Activated Parameters	37B	32.6B	↓ 13%
Experts (total)	256	384	↑ 50%
Experts Active per Token	8	8	=
Shared Experts	1	1	=
Attention Heads	128	64	↓ 50%
Number of Dense Layers	3	1	↓ 67%
Expert Grouping	Yes	No	-

Sparsity Scaling Law We develop a sparsity scaling law tailored for the Mixture-of-Experts (MoE) model family using Muon. Sparsity is defined as the ratio of the total number of experts to the number of activated experts. Through carefully controlled small-scale experiments, we observe that — under a fixed number of activated parameters (i.e., constant FLOPs) — increasing the total number of experts (i.e., increasing sparsity) consistently lowers both the training and validation loss, thereby enhancing overall model performance (Figure 5). Concretely, under the compute-optimal sparsity scaling law, achieving the same validation loss of 1.5, sparsity 48 reduces FLOPs by 1.69×, 1.39×, and 1.15× compared to sparsity levels 8, 16, and 32, respectively. Though increasing sparsity leads to better performance, this gain comes with increased infrastructure complexity. To balance model performance with cost, we adopt a sparsity of 48 for Kimi K2, activating 8 out of 384 experts per forward pass.

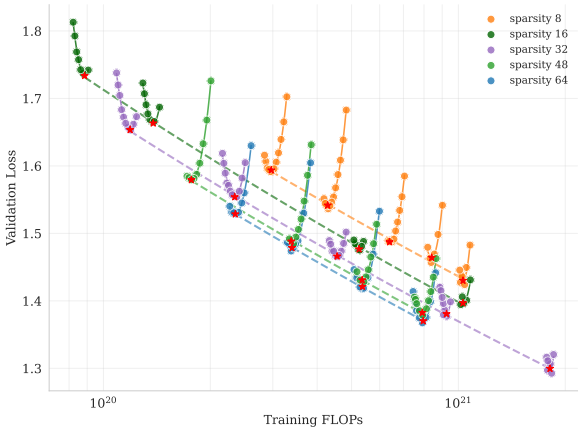


Figure 5: Sparsity Scaling Law. Increasing sparsity leads to improved model performance. We fixed the number of activated experts to 8 and the number of shared experts to 1, and varied the total number of experts, resulting in models with different sparsity levels.

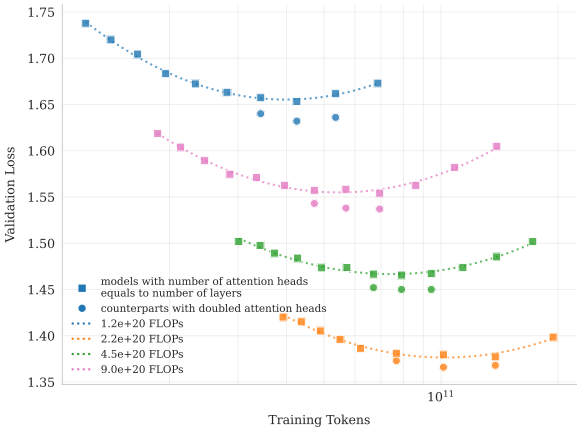


Figure 6: Scaling curves for models with number of attention heads equals to number of layers and their counterparts with doubled attention heads. Doubling the number of attention heads leads to a reduction in validation loss of approximately 0.5% to 1.2%.

Number of Attention Heads DeepSeek-V3 [10] sets the number of attention heads to roughly twice the number of model layers to better utilize memory bandwidth and enhance computational efficiency. However, as the context length increases, doubling the number of attention heads leads to significant inference overhead, reducing efficiency at longer sequence lengths. This becomes a major limitation in agentic applications, where efficient long context processing is essential. For example, with a sequence length of 128k, increasing the number of attention heads from 64 to 128, while keeping the total expert count fixed at 384, leads to an 83% increase in inference FLOPs. To evaluate the impact of this design, we conduct controlled experiments comparing configurations where the number of attention heads equals the number of layers against those with double number of heads, under varying training FLOPs. Under iso-token training conditions, we observe that doubling the attention heads yields only modest improvements in validation loss (ranging from 0.5% to 1.2%) across different compute budgets (Figure 6). Given that sparsity 48 already offers strong performance, the marginal gains from doubling attention heads do not justify the inference cost. Therefore we choose to 64 attention heads.

2.4 Training Infrastructure

2.4.1 Compute Cluster

Kimi K2 was trained on a cluster equipped with NVIDIA H800 GPUs. Each node in the H800 cluster contains 2 TB RAM and 8 GPUs connected by NVLink and NVSwitch within nodes. Across different nodes, 8×400 Gbps RoCE interconnects are utilized to facilitate communications.

2.4.2 Parallelism for Model Scaling

Training of large language models often progresses under dynamic resource availability. Instead of optimizing one parallelism strategy that's only applicable under specific amount of resources, we pursue a flexible strategy that allows Kimi K2 to be trained on any number of nodes that is a multiple of 32. Our strategy leverages a combination of 16-way

稀疏度扩展律　我们基于Muon 为 Mixture-of-Experts（MoE）模型族量身定制了一条稀疏度扩展律。稀疏度定义为专家总数与激活专家数之比。通过精心控制的小规模实验，我们观察到——在激活参数数量（即 FLOPs）固定的情况下——增加专家总数（即提高稀疏度）会持续降低训练与验证损失，从而提升整体模型性能（图 5）。具体而言，在计算最优的稀疏度扩展律下，要达到相同的验证损失 1.5，稀疏度 48 相比稀疏度 8、16、32 分别减少 1.69×、1.39×、1.15× 的 FLOPs。尽管提高稀疏度带来性能提升，但伴随而来的是基础设施复杂度的增加。为了在模型性能与成本之间取得平衡，我们在 Kimi K2 中采用稀疏度 48，每次前向传播激活 384 个专家中的 8 个。

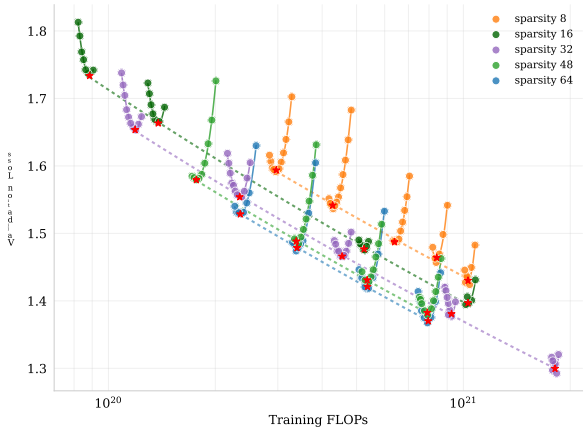


图5：稀疏性缩放定律。增加稀疏性可提升模型性能。我们固定激活专家数为8、共享专家数为1，并改变总专家数，从而得到不同稀疏度的模型。

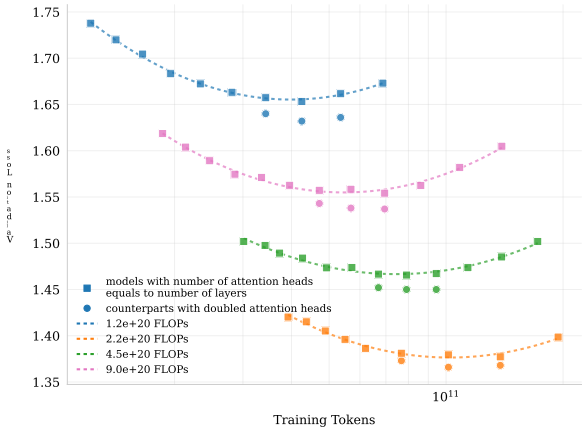


图6：注意力头数等于层数的模型及其注意力头数加倍的对应模型的扩展曲线。将注意力头数加倍可使验证损失降低约0.5%至1.2%。

注意力头数量　DeepSeek-V3 [10] 将注意力头数量设为模型层数的大约两倍，以更好地利用内存带宽并提升计算效率。然而，随着上下文长度增加，将注意力头数量翻倍会带来显著的推理开销，在更长序列长度下反而降低效率。这在代理应用中成为主要瓶颈，因为高效的长上下文处理至关重要。例如，当序列长度为 128 k 时，在总专家数固定为 384 的情况下，将注意力头数量从 64 增加到 128，会导致推理 FLOPs 增加 83%。为评估该设计的影响，我们在不同训练 FLOPs 下开展受控实验，比较注意力头数量等于层数的配置与头数翻倍的配置。在等 token 训练条件下，我们发现将注意力头翻倍仅带来验证损失的微小改善（在不同算力预算下提升 0.5% 至 1.2%，见图 6）。鉴于稀疏度 48 已能提供强劲性能，翻倍注意力头带来的边际收益不足以抵消推理成本。因此，我们选择 64 个注意力头。

2.4 训练基础设施

2.4.1 计算集群

Kimi K2 是在配备 NVIDIA H800 GPU 的集群上训练的。H800 集群中的每个节点包含 2 TB RAM 和 8 个通过 NVLink 和 NVSwitch 在节点内连接的 GPU。在不同节点之间，使用 8×400 Gbps RoCE 互连来促进通信。

2.4.2 模型扩展的并行性

大型语言模型的训练通常在动态资源可用性下进行。与其优化一种仅适用于特定资源量的并行策略，我们追求一种灵活的策略，使 Kimi K2 能够在任何为 32 的倍数的节点数上进行训练。我们的策略利用了 16 路

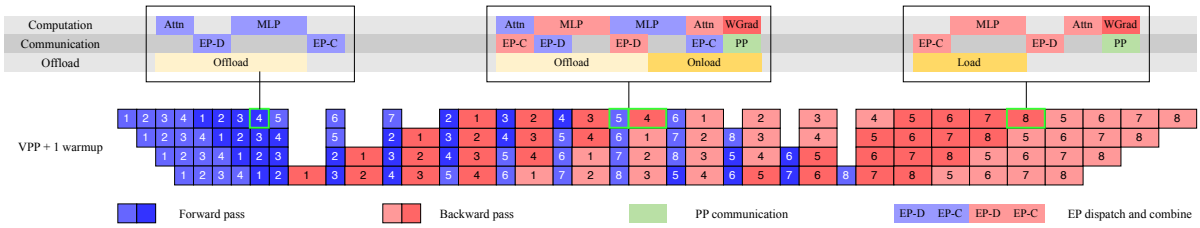


Figure 7: Computation, communication and offloading overlapped in different PP phases.

Pipeline Parallelism (PP) with virtual stages [28, 53, 38, 57, 47, 21], 16-way Expert Parallelism (EP) [39], and ZeRO-1 Data Parallelism [60].

Under this setting, storing the model parameters in BF16 and their gradient accumulation buffer in FP32 requires approximately 6 TB of GPU memory, distributed over a model-parallel group of 256 GPUs. Placement of optimizer states depends on the training configurations. When the total number of training nodes is large, the optimizer states are distributed, reducing its per-device memory footprint to a negligible level. When the total number of training nodes is small (e.g., 32), we can offload some optimizer states to CPU.

This approach allows us to reuse an identical parallelism configuration for both small- and large-scale experiments, while letting each GPU hold approximately 30 GB of GPU memory for all states. The rest of the GPU memory are used for activations, as described in Sec. 2.4.3. Such a consistent design is important for research efficiency, as it simplifies the system and substantially accelerates experimental iteration.

EP communication overlap with interleaved 1F1B By increasing the number of warm-up micro-batches, we can overlap EP all-to-all communication with computation under the standard interleaved 1F1B schedule [21, 53]. In comparison, DualPipe [10] doubles the memory required for parameters and gradients, necessitating an increase in parallelism to compensate. Increasing PP introduces more bubbles, while increasing EP, as discussed below, incurs higher overhead. The additional costs are prohibitively high for training a large model with over 1 trillion parameters and thus we opted not to use DualPipe.

However, interleaved 1F1B splits the model into more stages, introducing non-trivial PP communication overhead. To mitigate this cost, we decouple the weight-gradient computation from each micro-batch's backward pass and execute it in parallel with the corresponding PP communication. Consequently, all PP communications can be effectively overlapped except for the warm-up phase.

Smaller EP size To ensure full computation-communication overlap during the 1F1B stage, the reduced attention computation time in K2 (which has 64 attention heads compared to 128 heads in DeepSeek-V3) necessitates minimizing the time of EP operations. This is achieved by adopting the smallest feasible EP parallelization strategy, specifically EP = 16. Utilizing a smaller EP group also relaxes expert-balance constraints, allowing for near-optimal speed to be achieved without further tuning.

2.4.3 Activation Reduction

After reserving space for parameters, gradient buffers, and optimizer states, the remaining GPU memory on each device is insufficient to hold the full MoE activations. To ensure the activation memory fits within the constraints, especially for the initial pipeline stages that accumulate the largest activations during the 1F1B warm-up phase, the following techniques are employed.

Selective recomputation Recomputation is applied to inexpensive, high-footprint stages, including `LayerNorm`, `SwiGLU`, and `MLA` up-projections [10]. Additionally, `MoE` down-projections are recomputed during training to further reduce activation memory. While optional, this recomputation maintains adequate GPU memory, preventing crashes caused by expert imbalance in early training stages.

FP8 storage for insensitive activations Inputs of `MoE` up-projections and `SwiGLU` are compressed to FP8-E4M3 in $1\times$ 128 tiles with FP32 scales. Small-scale experiments show no measurable loss increase. Due to potential risks of performance degradation that we observed during preliminary study, we do not apply FP8 in computation.

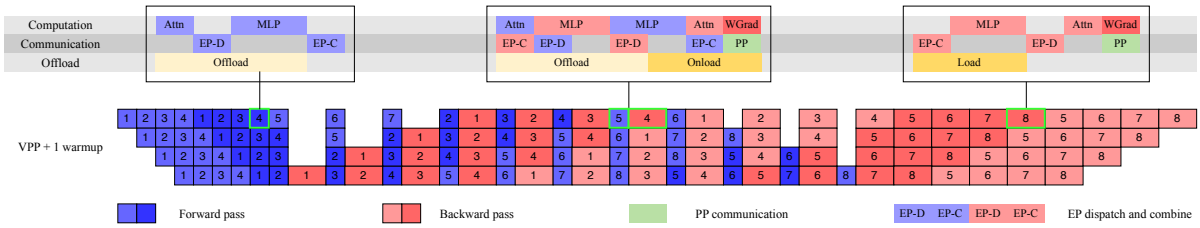


图7：在不同PP阶段中重叠的计算、通信与卸载

带有虚拟阶段的流水线并行（PP）[28, 53, 38, 57, 47, 21]、16路专家并行（EP）[39] 以及 ZeRO-1 数据并行 [60]。

在此设置下，将模型参数以 BF16 存储、其梯度累积缓冲区以 FP32 存储，大约需要 6 TB 的 GPU 显存，分布在 256 个 GPU 的模型并行组上。优化器状态的放置取决于训练配置。当训练节点总数较大时，优化器状态会被分布，使其在每个设备上的显存占用降至可忽略的水平。当训练节点总数较小（例如 32）时，我们可以将部分优化器状态卸载到 CPU。

这种方法使我们能够对小规模和大规模实验复用相同的并行配置，同时让每张 GPU 为所有状态保留约 30 GB 的显存。其余显存用于激活值，如第 2.4.3 节所述。这种一致的设计对研究效率至关重要，因为它简化了系统并显著加快了实验迭代。

通过增加预热微批次的数量，我们可以在标准交错 1F1B 调度 [21, 53] 下将 EP 的 all-to-all 通信与计算重叠。相比之下，DualPipe [10] 会使参数和梯度的内存需求翻倍，必须通过增加并行度来弥补。增加 PP 会引入更多气泡，而增加 EP（如下所述）则会带来更高开销。对于训练超过 1 万亿参数的大型模型，这些额外成本过高，因此我们选择不使用 DualPipe。

然而，交错式 1F1B 将模型切分为更多阶段，引入了不可忽视的 PP 通信开销。为缓解这一开销，我们将权重梯度计算从每个微批次的反向传播中解耦，并与对应的 PP 通信并行执行。因此，除预热阶段外，所有 PP 通信均可被有效重叠。

更小的 EP 规模　为了在 1F1B 阶段实现计算与通信的完全重叠，K2 中注意力计算时间的减少（K2 有 64 个注意力头，而 DeepSeek-V3 有 128 个）要求尽可能缩短 EP 操作的时间。为此，采用最小可行的 EP 并行策略，即 EP = 16。使用更小的 EP 组也放宽了专家负载均衡的约束，无需进一步调优即可接近最优速度。

2.4.3 激活减少

在为参数、梯度缓冲区和优化器状态预留空间后，每个设备上剩余的 GPU 内存不足以容纳完整的 MoE 激活。为了确保激活内存符合限制，尤其是在 1F1B 预热阶段累积最大激活的初始流水线阶段，采用了以下技术。

选择性重计算　重计算被应用于开销低但占用显存大的阶段，包括 LayerNorm、SwiGLU 和 MLA 上投影 {v*} [10]。此外，训练期间还会对 MoE 下投影进行重计算，以进一步减少激活显存。虽然可选，但此重计算能保持足够的 GPU 显存，防止早期训练阶段因专家负载不均而导致的崩溃。

FP8 存储用于不敏感激活值　MoE 上投影和 SwiGLU 的输入在 1× 128 个 tile 内被压缩为 FP8-E4M3，并附带 FP32 缩放因子。小规模实验显示损失没有可测量的增加。由于在初步研究中观察到潜在的性能下降风险，我们未在计算中使用 FP8。

Activation CPU offload All remaining activations are offloaded to CPU RAM. A copy engine is responsible for streaming the offload and onload, overlapping with both computation and communication kernels. During the 1F1B phase, we offload the forward activations of the previous micro-batch while prefetching the backward activations of the next. The warm-up and cool-down phases are handled similarly and the overall pattern is shown in Figure 7. Although offloading may slightly affect EP traffic due to PCIe traffic congestion, our tests show that EP communication remains fully overlapped.

2.5 Training recipe

We pre-trained the model with a 4,096-token context window using the MuonClip optimizer (Algorithm 1) and the WSD learning rate schedule [25], processing a total of 15.5T tokens. The first 10T tokens were trained with a constant learning rate of 2e-4 after a 500-step warm-up, followed by 5.5T tokens with a cosine decay from 2e-4 to 2e-5. Weight decay was set to 0.1 throughout, and the global batch size was held at 67M tokens. The overall training curve is shown in Figure 3.

Towards the end of pre-training, we conducted an annealing phase followed by a long-context activation stage. The batch size was kept constant at 67M tokens, while the learning rate was decayed from 2e-5 to 7e-6. In this phase, the model was trained on 400 billion tokens with a 4k sequence length, followed by an additional 60 billion tokens with a 32k sequence length. To extend the context window to 128k, we employed the YaRN method [55].

3 Post-Training

3.1 Supervised Fine-Tuning

We employ the Muon optimizer [33] in our post-training and recommend its use for fine-tuning with K2. This follows from the conclusion of our previous work [46] that a Muon-pre-trained checkpoint produces the best performance with Muon fine-tuning.

We construct a large-scale instruction-tuning dataset spanning diverse domains, guided by two core principles: maximizing prompt diversity and ensuring high response quality. To this end, we develop a suite of data generation pipelines tailored to different task domains, each utilizing a combination of human annotation, prompt engineering, and verification processes. We adopt K1.5 [35] and other in-house domain-specialized expert models to generate candidate responses for various tasks, followed by LLMs or human-based judges to perform automated quality evaluation and filtering. For agentic data, we create a data synthesis pipeline to teach models tool-use capabilities through multi-step, interactive reasoning.

3.1.1 Large-Scale Agentic Data Synthesis for Tool Use Learning

A critical capability of modern LLM agents is their ability to autonomously use unfamiliar tools, interact with external environments, and iteratively refine their actions through reasoning, execution, and error correction. Agentic tool use capability is essential for solving complex, multi-step tasks that require dynamic interaction with real-world systems. Recent benchmarks such as ACEBench [6] and τ-bench [85] have highlighted the importance of comprehensive tool-use evaluation, while frameworks like ToolLLM [58] and ACEBench [6] have demonstrated the potential of teaching models to use thousands of tools effectively.

However, training such capabilities at scale presents a significant challenge: while real-world environments provide rich and authentic interaction signals, they are often difficult to construct at scale due to cost, complexity, privacy and accessibility constraints. Recent work on synthetic data generation (AgentInstruct [51]; Self-Instruct [75]; StableToolBench [20]; ZeroSearch [66]) has shown promising results in creating large-scale data without relying on real-world interactions. Building on these advances and inspired by ACEBench [6]'s comprehensive data synthesis framework, we developed a pipeline that simulates real-world tool-use scenarios at scale, enabling the generation of tens of thousands of diverse and high-quality training examples.

There are three stages in our data synthesis pipeline, depicted in Fig. 8.

- *Tool spec generation*: we first construct a large repository of tool specs from both real-world tools and LLM-synthetic tools;
- *Agent and task generation*: for each tool-set sampled from the tool repository, we generate an agent to use the toolset and some corresponding tasks;
- *Trajectory generation*: for each agent and task, we generate trajectories where the agent finishes the task by invoking tools.

激活 CPU 卸载　所有剩余激活被卸载到CPU RAM。一个复制引擎负责流式卸载与加载，并与计算和通信内核重叠。在 1F1B 阶段，我们在预取下一个微批次反向激活的同时，卸载前一个微批次的前向激活。预热和冷却阶段以类似方式处理，整体模式如图 7 所示。尽管卸载可能因 PCIe 流量拥塞而轻微影响 EP 流量，我们的测试表明 EP 通信仍保持完全重叠。

2.5 训练方案

我们使用 MuonClip 优化器（算法 1）和 WSD 学习率调度器 [25]，以 4,096 个 token 的上下文窗口对模型进行了预训练，共处理了 15.5T 个 token。前 10T 个 token 在 500 步预热后使用 2e-4 的恒定学习率进行训练，随后 5.5T 个 token 采用从 2e-4 到 2e-5 的余弦衰减。权重衰减始终设为 0.1，全局批次大小保持在 67M 个 token。整体训练曲线如图 3 所示。

在预训练接近尾声时，我们进行了退火阶段，随后进入长上下文激活阶段。批次大小保持在 67M tokens 不变，学习率从 2e-5 衰减到 7e-6。在此阶段，模型先用 4k 序列长度训练了 4000 亿 tokens，再用 32k 序列长度额外训练了 600 亿 tokens。为了将上下文窗口扩展到 128k，我们采用了 YaRN 方法 [55]。

3 后训练

3.1 监督微调

我们在后训练中使用 Muon 优化器 [33]，并建议在使用 K2 进行微调时也采用它。这源于我们先前工作 [46] 的结论：Muon 预训练的检查点配合 Muon 微调可获得最佳性能。

我们构建了一个覆盖多个领域的大规模指令调优数据集，遵循两大核心原则：最大化提示多样性并确保高响应质量。为此，我们开发了一套针对不同任务领域的数据生成流水线，每条流水线都结合了人工标注、提示工程和验证流程。我们采用 K1.5 [35] 以及其他内部领域专用专家模型，为各类任务生成候选响应，随后由大语言模型或人工评审进行自动质量评估与过滤。对于智能体数据，我们设计了一条数据合成流水线，通过多步交互推理来教授模型使用工具的能力。

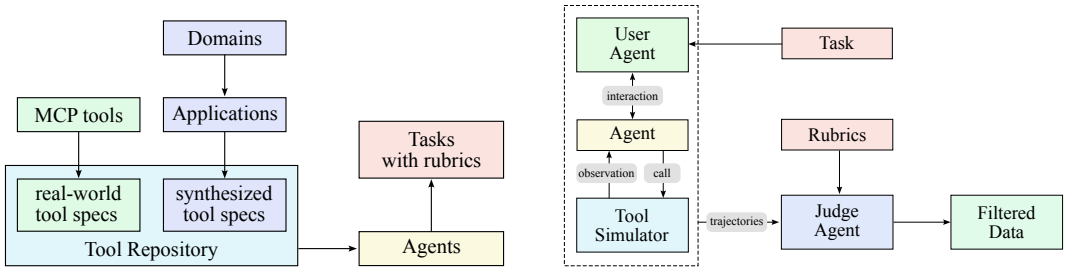
3.1.1 面向工具使用学习的大规模智能体数据合成

现代 LLM 智能体的一项关键能力，是能够自主使用陌生工具、与外部环境交互，并通过推理、执行和纠错迭代优化自身行为。具备智能体工具使用能力对于解决需要与现实世界系统动态交互的复杂多步任务至关重要。近期基准如 ACEBench [6] 和 τ-bench [85] 已凸显全面评估工具使用的重要性，而 ToolLLM [58] 和 ACEBench [6] 等框架则展示了教会模型有效使用数千种工具的潜力。

然而，大规模训练这类能力面临重大挑战：尽管真实世界环境能提供丰富且真实的交互信号，但由于成本、复杂性、隐私和可访问性等限制，往往难以大规模构建。近期在合成数据生成方面的工作（AgentInstruct [51]；Self-Instruct [75]；StableToolBench [20]；ZeroSearch [66]）已显示出在不依赖真实世界交互的情况下创建大规模数据的前景。基于这些进展，并受ACEBench [6]全面数据合成框架的启发，我们开发了一条流水线，可大规模模拟真实世界的工具使用场景，从而生成数万条多样且高质量的训练示例。

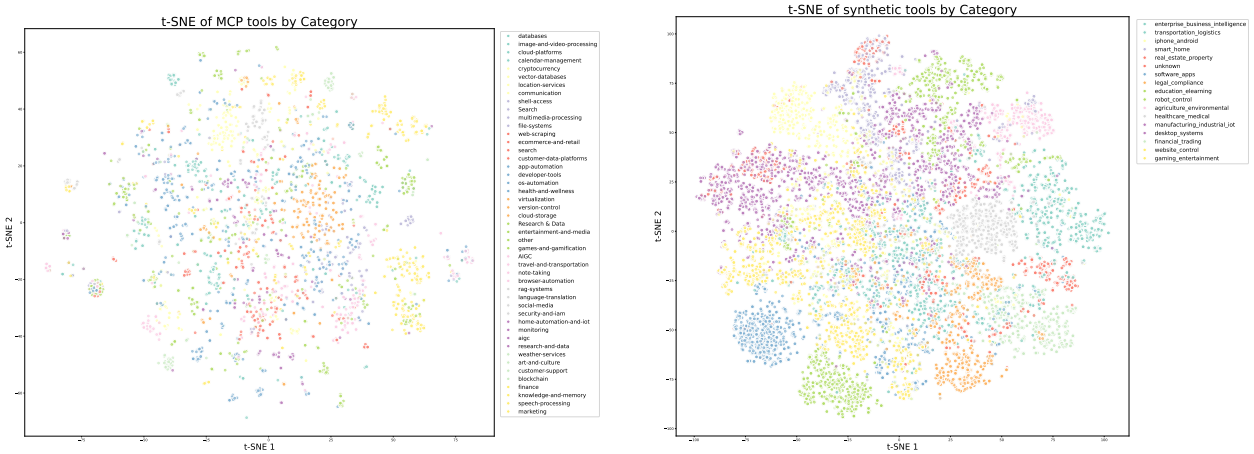
我们的数据合成流程包含三个阶段，如图8所示。

- *Tool spec generation*我们首先从真实世界工具和LLM合成工具中构建一个庞大的工具规范库；
- *Agent and task generation*对于从工具库中采样的每个工具集，我们生成一个代理来使用该工具集以及一些相应的任务；
- *Trajectory generation*：对于每个智能体和任务，我们生成轨迹，其中智能体通过调用工具完成任务。



(a) Synthesizing tool specs, agents and tasks (b) Generating agent trajectories

Figure 8: Data synthesis pipeline for tool use. (a) Tool specs are from both real-world tools and LLMs; agents and tasks are the generated from the tool repo. (b) Multi-agent pipeline to generate and filter trajectories with tool calling.



(a) t-SNE visualization of real MCP tools, colored by their original source categories

(b) t-SNE visualization of synthetic tools, colored by pre-defined domain categories

Figure 9: t-SNE visualizations of tool embeddings. (a) Real-world MCP tools exhibit natural clustering based on their original source categories. (b) Synthetic tools are organized into pre-defined domain categories, providing systematic coverage of the tool space. Together, they ensure comprehensive representation across different tool functionalities.

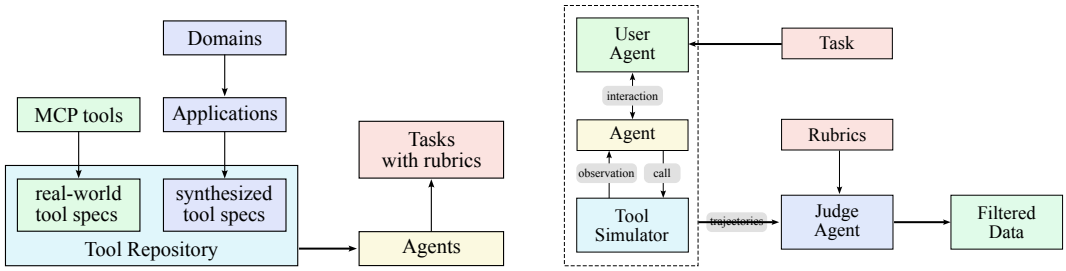
Domain Evolution and Tool Generation. We construct a comprehensive tool repository through two complementary approaches. First, we directly fetch 3000+ real MCP (Model Context Protocol) tools from GitHub repositories, leveraging existing high-quality tool specs. Second, we systematically evolve [82] synthetic tools through a hierarchical domain generation process: we begin with key categories (e.g., financial trading, software applications, robot control), then evolve multiple specific application domains within each category. Specialized tools are then synthesized for each domain, with clear interfaces, descriptions, and operational semantics. This evolution process produces over 20,000 synthetic tools. Figure 9 visualizes the diversity of our tool collection through t-SNE embeddings, demonstrating that both MCP and synthetic tools cover complementary regions of the tool space.

Agent Diversification. We generate thousands of distinct agents by synthesizing various system prompts and equipping them with different combinations of tools from our repository. This creates a diverse population of agents with varied capabilities, areas of expertise, and behavioral patterns, ensuring a broad coverage of potential use cases.

Rubric-Based Task Generation. For each agent configuration, we generate tasks that range from simple to complex operations. Each task is paired with an explicit rubric that specifies success criteria, expected tool-use patterns, and evaluation checkpoints. This rubric-based approach ensures a consistent and objective evaluation of agent performance.

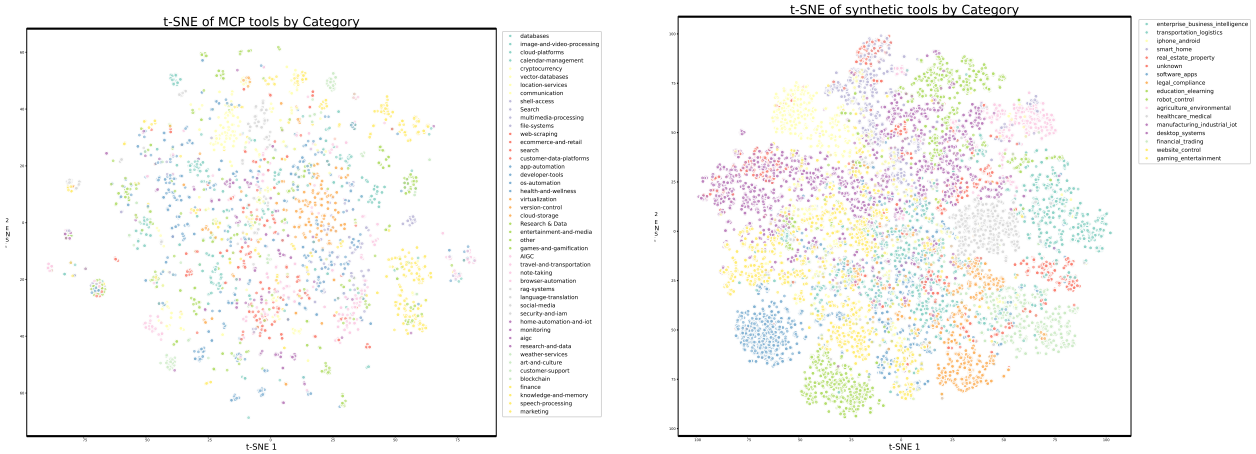
Multi-turn Trajectory Generation. We simulate realistic tool-use scenarios through several components:

- *User Simulation*: LLM-generated user personas with distinct communication styles and preferences engage in multi-turn dialogues with agents, creating naturalistic interaction patterns.



(a) Synthesizing tool specs, agents and tasks (b) Generating agent trajectories

图8：工具使用的数据合成流程。(a) 工具规范来自真实世界工具和LLM；智能体和任务由工具仓库生成。(b) 多智能体流程用于生成并过滤带有工具调用的轨迹。



(a) 真实 MCP 工具的 t-SNE 可视化，按原始来源类别着色 (b) 合成工具的 t-SNE 可视化，按预定义领域类别着色

图 9：工具嵌入的 t-SNE 可视化。(a) 真实世界的 MCP 工具根据其原始来源类别自然聚类。(b) 合成工具被组织到预定义的域类别中，系统性地覆盖工具空间。二者共同确保了对不同工具功能的全面表示。

领域演化与工具生成。我们通过两种互补的方法构建了一个全面的工具库。首先，我们直接从 GitHub 仓库获取 3000+ 个真实的 MCP（Model Context Protocol）工具，利用现有的高质量工具规范。其次，我们通过层次化的领域生成过程系统地演化 [82] 合成工具：从关键类别（如金融交易、软件应用、机器人控制）开始，然后在每个类别中演化多个具体的应用领域。接着为每个领域合成专用工具，配备清晰的接口、描述和操作语义。这一演化过程生成了超过 20,000 个合成工具。图 9 通过 t-SNE 嵌入可视化展示了我们工具集的多样性，表明 MCP 工具和合成工具覆盖了工具空间中互补的区域。

智能体多样化。我们通过合成各种系统提示，并为它们配备来自我们仓库的不同工具组合，生成数千个不同的智能体。这创建了一个多样化的智能体群体，具备不同的能力、专业领域和行为模式，确保对潜在用例的广泛覆盖。

基于评分标准的任务生成。对于每种智能体配置，我们生成从简单到复杂操作的任务。每个任务都配有一个明确的评分标准，规定成功标准、预期的工具使用模式和评估检查点。这种基于评分标准的方法确保了对智能体性能的一致且客观的评估。

多轮轨迹生成。我们通过多个组件模拟真实的工具使用场景：

- *User Simulation*：由LLM生成的具有不同沟通风格和偏好的用户角色与智能体进行多轮对话，创造出自然的交互模式。

- *Tool Execution Environment*: A sophisticated tool simulator (functionally equivalent to a world model) executes tool calls and provides realistic feedback. The simulator maintains and updates state after each tool execution, enabling complex multi-step interactions with persistent effects. It introduces controlled stochasticity to produce varied outcomes including successes, partial failures, and edge cases.

Quality Evaluation and Filtering. An LLM-based judge evaluates each trajectory against the task rubrics. Only trajectories that meet the success criteria are retained for training, ensuring high-quality data while allowing natural variation in task-completion strategies.

Hybrid Approach with Real Execution Environments. While simulation provides scalability, we acknowledge the inherent limitation of simulation fidelity. To address this, we complement our simulated environments with real execution sandboxes for scenarios where authenticity is crucial, particularly in coding and software engineering tasks. These real sandboxes execute actual code, interact with genuine development environments, and provide ground-truth feedback through objective metrics such as test suite pass rates. This combination ensures that our models learn from both the diversity of simulated scenarios and the authenticity of real executions, significantly strengthening practical agent capabilities.

By leveraging this hybrid pipeline that combines scalable simulation with targeted real-world execution, we generate diverse, high-quality tool-use demonstrations that balance coverage and authenticity. The scale and automation of our synthetic data generation, coupled with the grounding provided by real execution environments, effectively implements large-scale rejection sampling [26, 87] through our quality filtering process. This high-quality synthetic data, when used for supervised fine-tuning, has demonstrated significant improvements in the model's tool-use capabilities across a wide range of real-world applications.

3.2 Reinforcement Learning

Reinforcement learning (RL) is believed to have better token efficiency and generalization than SFT. Based on the work of K1.5 [35], we continue to scale RL in both task diversity and training FLOPs in K2. To support this, we develop a Gym-like extensible framework that facilitates RL across a wide range of scenarios. We extend the framework with a large number of tasks with verifiable rewards. For tasks that rely on subjective preferences, such as creative writing and open-ended question answering, we introduce a self-critic reward in which the model performs pairwise comparisons to judge its own outputs. This approach allows tasks from various domains to all benefit from the RL paradigm.

3.2.1 Verifiable Rewards Gym

Math, STEM and Logical Tasks For math, stem and logical reasoning domains, our RL data preparation follows two key principles, *diverse coverage* and *moderate difficulty*.

Diverse Coverage. For math and stem tasks, we collect high-quality QA pairs using a combination of expert annotations, internal QA extraction pipelines, and open datasets [41, 52]. During the collection process, we leverage a tagging system to deliberately increase coverage of under-covered domains. For logical tasks, our dataset comprises a variety of formats, including structured data tasks (e.g., multi-hop tabular reasoning, cross-table aggregation) and logic puzzles (e.g., the 24-game, Sudoku, riddles, cryptarithms, and Morse-code decoding).

Moderate Difficulty. The RL prompt-set should be neither too easy nor too hard, both of which may produce little signal and reduce learning efficiency. We assess the difficulty of each problem using the SFT model's pass@k accuracy and select only problems with moderate difficulty.

Complex Instruction Following Effective instruction following requires not only understanding explicit constraints but also navigating implicit requirements, handling edge cases, and maintaining consistency over extended dialogues. We address these challenges through a hybrid verification framework that combines automated verification with adversarial detection, coupled with a scalable curriculum generation pipeline. Our approach employs a dual-path system to ensure both precision and robustness:

Hybrid Rule Verification. We implement two verification mechanisms: (1) deterministic evaluation via code interpreters for instructions with verifiable outputs (e.g., length, style constraints), and (2) LLM-as-judge evaluation for instructions requiring nuanced understanding of constraints. To address potential adversarial behaviors where models might claim instruction fulfillment without actual compliance, we incorporate an additional hack-check layer that specifically detects such deceptive claims.

Multi-Source Instruction Generation. To construct our training data, we employ three distinct generation strategies to ensure comprehensive coverage: (1) expert-crafted complex conditional prompts and rubrics developed by our data

- *Tool Execution Environment*：一个复杂的工具模拟器（功能上等价于世界模型）执行工具调用并提供逼真的反馈。该模拟器在每次工具执行后维护并更新状态，从而实现具有持久效果的复杂多步交互。它引入受控的随机性，以产生多样化的结果，包括成功、部分失败和边缘情况。

质量评估与过滤。基于 LLM 的评判器根据任务评分标准评估每条轨迹。仅保留满足成功标准的轨迹用于训练，在确保高质量数据的同时，允许任务完成策略存在自然变化。

混合方法结合真实执行环境。虽然模拟提供了可扩展性，我们仍承认模拟保真度的固有限制。为此，我们在模拟环境之外补充了真实执行沙箱，用于那些对真实性要求极高的场景，特别是编码和软件工程任务。这些真实沙箱执行实际代码，与真实的开发环境交互，并通过测试套件通过率等客观指标提供真实反馈。这种组合确保我们的模型既能从模拟场景的多样性中学习，也能从真实执行的真实性中获益，显著增强了智能体的实际能力。

通过结合可扩展仿真与针对性真实世界执行的混合流水线，我们生成了多样化、高质量的工具使用演示，兼顾覆盖度与真实性。我们合成数据生成的规模与自动化，加上真实执行环境提供的落地验证，通过质量过滤流程有效实现了大规模拒绝采样 [26, 87]。当这些高质量合成数据用于监督微调时，已在广泛的现实应用中显著提升了模型的工具使用能力。

3.2 强化学习

强化学习（RL）被认为在 token 效率和泛化能力上优于 SFT。基于 K1.5 [35] 的工作，我们在 K2 中继续在任务多样性和训练 FLOPs 两个维度上扩展 RL。为此，我们开发了一个类似 Gym 的可扩展框架，便于在广泛场景中应用 RL。我们通过大量带有可验证奖励的任务对该框架进行了扩展。对于依赖主观偏好的任务，如创意写作和开放式问答，我们引入了自批评奖励，让模型通过成对比较来评判自己的输出。这种方法使来自不同领域的任务都能从 RL 范式中受益。

3.2.1 可验证奖励健身房

数学、STEM 和逻辑任务　对于数学STEM 和逻辑推理领域，我们的 RL 数据准备遵循两个关键原则：*diverse coverage* 和 *moderate difficulty*。

Diverse Coverage. 对于数学和 STEM 任务，我们通过专家标注、内部 QA 提取流程以及开放数据集 [41, 52] 的组合来收集高质量的 QA 对。在收集过程中，我们利用标签系统刻意增加对覆盖不足领域的覆盖。对于逻辑任务，我们的数据集包含多种格式，包括结构化数据任务（例如多跳表格推理、跨表聚合）和逻辑谜题（例如 24 点游戏、数独、谜语、字母算术和摩斯密码解码）。

Moderate Difficulty. RL 提示集既不应过于简单，也不应过于困难，这两种情况都可能产生很少的信号并降低学习效率。我们使用 SFT 模型的 pass@k 准确率评估每个问题的难度，并仅选择难度适中的问题。

复杂指令遵循　有效的指令遵循不仅需要理解显式约束，还需处理隐含要求、应对边缘情况，并在长对话中保持一致性。我们通过混合验证框架应对这些挑战，该框架将自动验证与对抗检测相结合，并辅以可扩展的课程生成流水线。我们的方法采用双路径系统，以确保精确性与鲁棒性并重：

Hybrid Rule Verification. 我们实现了两种验证机制：（1）通过代码解释器对具有可验证输出（例如长度、风格约束）的指令进行确定性评估，以及（2）对需要细致理解约束的指令采用 LLM-as-judge 评估。为了应对模型可能声称已遵循指令却实际未遵循的潜在对抗行为，我们增加了一个额外的 hack-check 层，专门检测此类欺骗性声明。

Multi-Source Instruction Generation. 为了构建训练数据，我们采用三种不同的生成策略以确保全面覆盖：(1) 由我们的数据团队开发、由专家精心设计的复杂条件提示和评分标准

team (2) agentic instruction augmentation inspired by AutoIF [12], and (3) a fine-tuned model specialized for generating additional instructions that probe specific failure modes or edge cases. This multipronged approach ensures both breadth and depth in instruction coverage.

Faithfulness Faithfulness is essential for an agentic model operating in scenarios such as multi-turn tool use, self-generated reasoning chains, and open-environment interactions. Inspired by the evaluation framework from FACTS Grounding [30], we train a sentence-level faithfulness judge model to perform automated verification. The judge is effective in detecting sentences that make a factual claim without supporting evidence in context. It serves as a reward model to enhance overall faithfulness performance.

Coding & Software Engineering To enhance our capability in tackling competition-level programming problems, we gather problems and their judges from both open-source datasets [27, 83] and synthetic sources. To ensure the diversity of the synthetic data and the correctness of reward signals, we incorporate high-quality human-written unit tests retrieved from pre-training data.

For software engineering tasks, we collect a vast amount of pull requests and issues from GitHub to build software development environment that consists of user prompts/issues and executable unit tests. This environment was built on a robust sandbox infrastructure, powered by Kubernetes for scalability and security. It supports over 10,000 concurrent sandbox instances with stable performance, making it ideal for both competitive coding and software engineering tasks.

Safety Our work to enhance the safety begins with a human-curated set of seed prompts, manually crafted to encompass prevalent risk categories such as violence, fraud, and discrimination.

To simulate sophisticated jailbreak attempts (e.g., role-playing, literary narratives, and academic discourse), we employ an automated prompt evolution pipeline with three key components:

- **Attack Model**: Iteratively generates adversarial prompts designed to elicit unsafe responses from the target LLM.
- **Target Model**: Produces responses to these prompts, simulating potential vulnerabilities.
- **Judge Model**: Evaluates the interaction to determine if the adversarial prompt successfully bypasses safety mechanisms.

Each interaction is assessed using a task-specific rubric, enabling the judge model to provide a binary success/failure label.

3.2.2 Beyond Verification: Self-Critique Rubric Reward

To extend model alignment beyond tasks with verifiable reward, we introduce a framework for general reinforcement learning from self-critic feedbacks. This approach is designed to align LLMs with nuanced human preferences, including helpfulness, creativity, depth of reasoning, factuality, and safety, by extending the capabilities learned from verifiable scenarios to a broader range of subjective tasks. The framework operates using a *Self-Critique Rubric Reward* mechanism, where the model evaluates its own outputs to generate preference signals. To bootstrap K2 as a competent judge, we curated a mixture of open-source and in-house preference datasets and initialize its critic capability in the SFT stage.

Self-Critiqued Policy Optimization In the first core process of the learning loop, the K2 actor generates responses for general prompts that cover a wide range of use cases. The K2 critic then ranks all results by performing pairwise evaluations against a combination of rubrics, which incorporates both *core rubrics* (Appendix. F.1), which represent the fundamental values of our AI assistant that Kimi cherish, prescriptive rubrics (Appendix. F.2) that aim to eliminate reward hacking, and *human-annotated rubrics* crafted by our data team for specific instructional contexts. Although certain rubrics can be designated as mandatory, K2 retains the flexibility to weigh them against its internal priors. This capacity enables a dynamic and continuous alignment with its evolving on-policy behavior, ensuring that the model's responses remain coherent with its core identity while adapting to specific instructions.

Closed-Loop Critic Refinement and Alignment During RL training, the critic model is refined using verifiable signals. On-policy rollouts generated from verifiable-reward prompts are used to continuously update the critic, a crucial step that distills objective performance signals from RLVR directly into its evaluation model. This transfer learning process grounds its more subjective judgments in verifiable data, allowing the performance gains from verifiable tasks to enhance the critic's judgment on complex tasks that lack explicit reward signals. This closed-loop process ensures that the critic continuously recalibrates its evaluation standards in lockstep with the policy's evolution. By

团队（2）受 AutoIF [12] 启发的代理式指令增强，以及（3）一个经过微调的模型，专门用于生成额外指令，以探测特定的失败模式或边缘情况。这种多管齐下的方法确保了指令覆盖的广度与深度。

忠实性 对于在多轮工具使用、自生成推理链和开放环境交互等场景中运行的智能体模型而言，忠实性至关重要。受 FACTS Grounding [30] 评估框架的启发，我们训练了一个句子级忠实性裁判模型，用于自动验证。该裁判能有效检测出在上下文中缺乏支持证据却提出事实性断言的句子。它作为奖励模型，用于提升整体的忠实性表现。

编码与软件工程 为了提升我们解决竞赛级编程问题的能力，我们从开源数据集 [27, 83] 和合成来源收集问题及其评测器。为了确保合成数据的多样性以及奖励信号的正确性，我们引入了从预训练数据中检索的高质量人工编写的单元测试。

对于软件工程任务，我们从 GitHub 收集了大量拉取请求和问题，以构建由用户提示/问题和可执行单元测试组成的软件开发环境。该环境建立在强大的沙箱基础设施之上，利用 Kubernetes 实现可扩展性和安全性。它支持超过 10,000 个并发沙箱实例，性能稳定，非常适合竞技编程和软件工程任务。

安全 我们提升安全性的工作始于由人工策划的一组种子提示，这些提示经过精心设计，涵盖了暴力、欺诈和歧视等常见风险类别。

为了模拟复杂的越狱尝试（例如角色扮演、文学叙事和学术论述），我们采用了一个包含三个关键组件的自动化提示演化流程：

- 攻击模型：迭代生成旨在诱导目标LLM产生不安全回复的对抗性提示。
- 目标模型：对这些提示生成响应，模拟潜在漏洞。
- 评判模型：评估交互以判断对抗性提示是否成功绕过安全机制。

每次交互都使用特定任务的评分标准进行评估，使评判模型能够提供二元成功/失败标签。

3.2.2 超越验证：自我批判评分奖励

为了将模型对齐扩展到具有可验证奖励之外的任务，我们引入了一个基于自我批判反馈的通用强化学习框架。该方法旨在通过将可验证场景中学到的能力扩展到更广泛的主观任务，使大语言模型与人类在有用性、创造力、推理深度、事实性和安全性等方面的细微偏好保持一致。该框架采用*Self-Critique Rubric Reward*机制，模型通过评估自身输出来生成偏好信号。为了让K2成为一个合格的评判者，我们策划了开源与内部偏好数据集的混合，并在SFT阶段初始化其批判能力。

自我批判的策略优化 在学习循环的第一个核心过程中，K2 行动者针对涵盖广泛用例的通用提示生成回复。随后，K2 批判者通过成对评估对所有结果进行排序，评估依据是一套综合评分标准，其中包括 *core rubrics* (附录 F.1)（代表我们珍视的 AI 助手的基本价值观）、旨在消除奖励劫持的规范性评分标准（附录 F.2），以及由我们的数据团队针对特定指令情境精心设计的 *human-annotated rubrics*。尽管某些评分标准可被指定为强制，K2 仍保留根据内部先验对其进行权衡的灵活性。这一能力使其能够与其不断演化的在策略行为保持动态且持续的契合，确保模型的回复在遵循核心身份的同时，也能适应具体指令。

闭环评论家精炼与对齐 在RL 训练期间，评论家模型通过可验证信号进行精炼。利用来自可验证奖励提示的 on-policy 轨迹持续更新评论家，这是将 RLVR 中的客观性能信号直接蒸馏到其评估模型的关键步骤。该迁移学习过程将其更主观的判断锚定在可验证数据上，使可验证任务带来的性能提升能够增强评论家对缺乏显式奖励信号的复杂任务的判断。这一闭环过程确保评论家持续重新校准其评估标准，与策略的演进保持同步。

grounding subjective evaluation in verifiable data, the framework enables robust and scalable alignment with complex, non-verifiable human objectives.

Consequently, this holistic alignment yields comprehensive performance improvements across a wide spectrum of domains, including user intent understanding, creative writing, complex reasoning, and nuanced language comprehension.

3.2.3 RL Algorithm

We adopt the policy optimization algorithm introduced in K1.5 [35] as the foundation for K2. For each problem x, we sample K responses $\{y_1, \ldots, y_k\}$ from the previous policy π_{old}, and optimize the model π_θ with respect to the following objective:

$$L_{\text{RL}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[\frac{1}{K} \sum_{i=1}^{K} \left[\left(r(x, y_i) - \bar{r}(x) - \tau \log \frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)} \right)^2 \right] \right],$$

where $\bar{r}(x) = \frac{1}{k} \sum_{i=1}^{k} r(x, y_i)$ is the mean rewards of the sampled responses, $\tau > 0$ is a regularization parameter that promotes stable learning. As in SFT, we employ the Muon optimizer [33] to minimize this objective. As we scale RL training to encompass a broader range of tasks in K2, a primary challenge is achieving consistent performance improvements across all domains. To address this, we introduce several additions to the RL algorithm.

Budget Control It has been widely observed that RL often results in a substantial increase in the length of model-generated responses [35, 19]. While longer responses can enable the model to utilize additional test-time compute for improved performance on complex reasoning tasks, the benefits often do not justify its inference cost in non-reasoning domains. To encourage the model to properly distribute inference budget, we enforce a per-sample *maximum token budget* throughout RL training, where the budget is determined based on the type of task. Responses that exceed this token budget are truncated and assigned a penalty, which incentivizes the model to generate solutions within the specified limit. Empirically, this approach significantly enhances the model's token efficiency, encouraging concise yet effective solutions across all domains.

PTX Loss To prevent the potential forgetting of valuable, high-quality data during joint RL training, we curate a dataset comprising hand-selected, high-quality samples and integrate it into the RL objective through an auxiliary PTX loss [54]. This strategy not only leverages the advantages of high-quality data, but also mitigates the risk of overfitting to the limited set of tasks explicitly present in the training regime. This augmentation substantially improves the model's generalization across a broader range of domains.

Temperature Decay For tasks such as creative writing and complex reasoning, we find that promoting exploration via a high sampling temperature during the initial stages of training is crucial. A high temperature allow the model to generate diverse and innovative responses, thereby facilitating the discovery of effective strategies and reducing the risk of premature convergence to suboptimal solutions. However, retaining a high temperature in the later stages of training or during evaluation can be detrimental, as it introduces excessive randomness and compromises the reliability and consistency of the model's outputs. To address this, we employ a temperature decay schedule, to shift from exploration to exploitation throughout the training. This strategy ensures that the model leverages exploration when it is most beneficial, while ultimately converge on stable and high-quality outputs.

3.3 RL Infrastructure

3.3.1 Colocated Architecture

Similar to K1.5 [35], we adopt a hybrid colocated architecture for our synchronized RL training, where the training and inference engines live on the same workers. When one engine is actively working, the other engine releases or offloads its GPU resources to accommodate. In each iteration of RL training, a centralized controller first calls the inference engine to generate new data for training. It then notifies the training engine to train on the new data, and send updated parameters to the inference engine for the next iteration.

Each engine is heavily optimized for throughput. In addition, as the model scales to the size of K2, the latency of engine switching and failure recovery becomes significant. We present our system design considerations in these aspects.

通过将主观评估建立在可验证的数据基础上，该框架实现了与复杂、不可验证的人类目标的稳健且可扩展的对齐。

因此，这种整体对齐在广泛的领域中带来了全面的性能提升，包括用户意图理解、创意写作、复杂推理和细致语言理解。

3.2.3 RL算法

我们采用 K1.5 [35] 中引入的策略优化算法作为 K2 的基础。对于每个问题 x，我们从先前策略 π_{old} 中采样 K 个响应 $\{y_1, \ldots, y_k\}$，并针对以下目标优化模型 π_θ：

$$L_{\text{RL}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[\frac{1}{K} \sum_{i=1}^{K} \left[\left(r(x, y_i) - \bar{r}(x) - \tau \log \frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)} \right)^2 \right] \right],$$

其中 $\bar{r}(x) = \frac{1}{k} \sum_{i=1}^{k} r(x, y_i)$ 是采样响应的平均奖励，$\tau > 0$ 是一个促进稳定学习的正则化参数。与 SFT 一样，我们使用 Muon 优化器 [33] 来最小化该目标。随着我们将 RL 训练扩展到 K2 中更广泛的任务，一个主要挑战是在所有领域实现一致的性能提升。为此，我们对 RL 算法引入了几项补充。

预算控制 已有广泛观察表明，强化学习RL）通常会导致模型生成回复的长度显著增加 [35, 19]。虽然更长的回复可以让模型在复杂推理任务中利用额外的测试时计算以提升性能，但在非推理领域，其带来的收益往往无法抵消推理成本。为了鼓励模型合理分配推理预算，我们在整个 RL 训练过程中对每个样本强制执行 *maximum token budget*，预算根据任务类型确定。超出该 token 预算的回复将被截断并受到惩罚，从而激励模型在指定限制内生成解决方案。实验表明，该方法显著提升了模型的 token 效率，在所有领域都鼓励了简洁而有效的解决方案。

PTX 损失 为防止在联合 RL 训练中遗忘有价值的高质量数据，我们精心构建了一个由人工挑选的高质量样本组成的数据集，并通过辅助 PTX 损失 [54] 将其纳入 RL 目标。该策略不仅充分利用了高质量数据的优势，还缓解了因训练体系中显式任务有限而导致的过拟合风险。这一增强显著提升了模型在更广泛领域中的泛化能力。

温度衰减 对于创意写作和复杂推理等任务，我们发现通过在训练初期采用高采样温度来促进探索至关重要。高温度使模型能够生成多样且创新的响应，从而有助于发现有效策略，并降低过早收敛到次优解的风险。然而，在训练后期或评估阶段仍保持高温度则可能有害，因为它会引入过多随机性，损害模型输出的可靠性与一致性。为此，我们采用温度衰减调度，在整个训练过程中实现从探索到利用的过渡。该策略确保模型在最需要时充分利用探索，最终收敛到稳定且高质量的输出。

3.3 RL 基础设施

3.3.1 同位架构

与 K1.5 [35] 类似，我们采用混合同址架构进行同步 RL 训练，训练引擎和推理引擎位于同一组工作节点上。当一个引擎处于活跃状态时，另一个引擎会释放或卸载其 GPU 资源以腾出空间。在每次 RL 训练迭代中，一个集中式控制器首先调用推理引擎生成新的训练数据，然后通知训练引擎在这些新数据上进行训练，并将更新后的参数发送给推理引擎，供下一次迭代使用。

每个引擎都针对吞吐量进行了高度优化。此外，随着模型扩展到 K2 规模，引擎切换和故障恢复的延迟变得显著。我们在此方面介绍了系统设计考量。

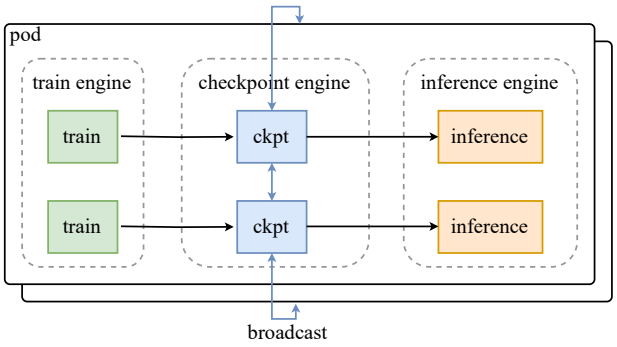


Figure 10: Parameter update utilizing a checkpoint engine

3.3.2 Efficient Engine Switching

During rollout, the parameters of the training engine are offloaded to DRAM. Bringing up the training engine is therefore a simple step of H2D transmission. However, bringing up the inference engine is a bigger challenge, as it must obtain updated parameters from the training engine with a different sharding paradigm.

Given the scale of K2 and the vast number of devices involved, using a network file system for resharding and broadcasting parameters is impractical. The aggregate bandwidth required to keep overhead low reaches several petabytes per second. To address this challenge, we developed a distributed checkpoint engine co-located on training nodes to manage parameter states. To perform a parameter update, each checkpoint engine worker obtains a local copy of parameters from the training engine, then broadcasts the full parameter set across all checkpoint engine workers. Subsequently, the inference engine retrieves only the parameter shard it requires from the checkpoint engine. This process is illustrated in Figure 10. To enable this for a 1T model, updates are performed parameter-by-parameter in a pipelined manner, minimizing memory footprint (see Appendix G).

We opt to broadcast the full parameter set across the entire cluster, regardless of the specific sharding schemes on each inference worker. While this transfers several times more data than a theoretically optimal approach, it offers a simpler system design that is less intrusive to the training and inference engines. We chose to trade off this minor overhead to fully decouple the training engine and the inference engine, significantly simplifying maintenance and testing.

Notably, this approach outperforms the transfer-what-you-need method due to reduced synchronization overhead and higher network bandwidth utilization. Our system can complete a full parameter update for Kimi K2 with less than 30 seconds, a negligible duration for a typical RL training iteration.

3.3.3 Efficient System Startup

As large-scale training is prone to system failure, optimizing the startup time is crucial for models as large as Kimi K2.

To start the training engine, we let each training worker selectively read part or none of the parameters from disk, and broadcast necessary parameters to its peers. The design goal is to ensure all workers collectively read the checkpoint only once, minimizing expensive disk IO.

As the inference engines are independent replicas, we would like to avoid introducing extra synchronization barriers between them. Therefore, we opt to reuse checkpoint engine for startup: we let checkpoint engine collectively read the checkpoint from disk, similar to how the training engine starts. Then it updates the state of the uninitialized inference engine, using the approach introduced in the previous section. By leveraging the dedicated checkpoint engine, the system also becomes robust to single-point failures, because an inference replica can restart without communicating with other replicas.

3.3.4 Agentic Rollout

Our RL infrastructure supports the training of long-horizon, multi-turn agentic tasks. During rollout, these tasks present distinct challenges, such as complex environmental interactions and prolonged rollout durations. Here we introduce a few optimizations to alleviate these issues.

Due to the diversity of environments, certain interactions may be blocked on waiting for environment feedback (e.g., a virtual machine or a code interpreter), leaving the GPUs idle. We employ two strategies to maximize GPU utilization:

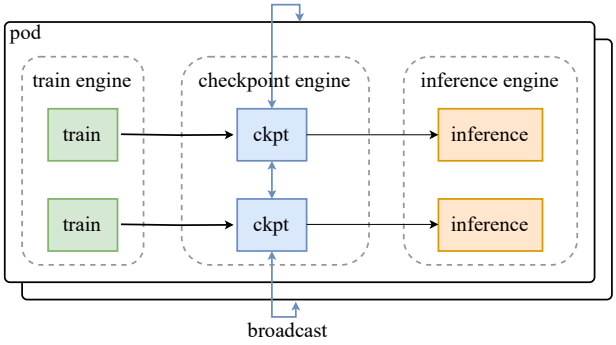


图10：利用检查点引擎的参数更新

3.3.2 高效引擎切换

在 rollout 过程中，训练引擎的参数被卸载到 DRAM。因此，启动训练引擎只是一个简单的 H2D 传输步骤。然而，启动推理引擎则是一个更大的挑战，因为它必须从训练引擎获取更新后的参数，而两者的分片范式不同。

鉴于 K2 的规模以及所涉及的庞大设备数量，使用网络文件系统进行重分片和广播参数是不切实际的。为了将开销保持在较低水平，所需的聚合带宽达到每秒数 PB。为了解决这一挑战，我们在训练节点上共同部署了一个分布式检查点引擎来管理参数状态。为了执行参数更新，每个检查点引擎工作进程从训练引擎获取参数的本地副本，然后将完整参数集广播给所有检查点引擎工作进程。随后，推理引擎仅从检查点引擎检索其所需的参数分片。该过程如图 10 所示。为了对 1T 模型启用此功能，更新以流水线方式逐个参数进行，从而最小化内存占用（见附录 G）。

我们选择在整个集群中广播完整的参数集，而不考虑每个推理工作节点上的具体分片方案。虽然这会比理论上最优的方法多传输数倍的数据，但它带来了更简单的系统设计，对训练和推理引擎的侵入性更小。我们选择用这一微小开销来换取训练引擎与推理引擎的完全解耦，从而显著简化了维护和测试。

值得注意的是，由于同步开销更低且网络带宽利用率更高，该方法优于“按需迁移”策略。我们的系统可在不到 30 秒内完成 Kimi K2 的完整参数更新，这一时长对于典型的 RL 训练迭代而言可以忽略不计。

3.3.3 高效的系统启动

由于大规模训练容易出现系统故障，对于像 Kimi K2 这样大的模型，优化启动时间至关重要。

为了启动训练引擎，我们让每个训练工作进程从磁盘选择性地读取部分或全部参数，并将必要的参数广播给对等节点。设计目标是确保所有工作进程总共只读取一次检查点，从而将昂贵的磁盘 I/O 降至最低。

由于推理引擎是独立的副本，我们希望避免在它们之间引入额外的同步屏障。因此，我们选择复用检查点引擎来启动：我们让检查点引擎像训练引擎启动时那样，从磁盘集体读取检查点。然后，它使用上一节介绍的方法更新未初始化的推理引擎的状态。通过利用专用的检查点引擎，系统也对单点故障具有鲁棒性，因为推理副本可以在不与其他副本通信的情况下重启。

3.3.4 智能体部署

我们的 RL 基础设施支持长周期、多轮智能体任务的训练。在 rollout 过程中，这些任务带来了独特的挑战，例如复杂的环境交互和延长的 rollout 时长。在此我们介绍几项优化措施以缓解这些问题。

由于环境的多样性，某些交互可能会因等待环境反馈（例如虚拟机或代码解释器）而被阻塞，导致 GPU 空闲。我们采用两种策略来最大化 GPU 利用率：

(i) we deploy heavy environments as dedicated services that can scale up more easily; (ii) we employ a large number of concurrent rollouts to amortize the latency induced by certain expensive interactions.

Another challenge in agentic rollout is that individual rollout trajectories can be extremely long. To prevent long-tail trajectories from blocking the entire rollout process, we employ the partial rollout [35] technique. This strategy allows long-tail unfinished tasks to be paused, and resumed in the next RL iteration.

To improve research efficiency, we also design a unified interface inspired by the OpenAI Gym framework [49] to streamline the integration of new environments. We hope to scale our RL infrastructure to more diverse interactive environments in the future.

4 Evaluations

This section begins with the post-training evaluation of Kimi-K2-Instruct, followed by a brief overview of the capabilities of Kimi-K2-Base. We conclude with a comprehensive safety evaluation.

4.1 Post-training Evaluations

4.1.1 Evaluation Settings

Benchmarks We assess Kimi-K2-Instruct across different areas. For coding, we adopt LiveCodeBench v6 [31](questions from August 2024 to May 2025), OJBench [77], MultiPL-E [5], SWE-bench Verified [32, 84], TerminalBench [71], Multi-SWE-bench [86], SWE-Lancer [50], PaperBench [65], and Aider-Polyglot [16]. For tool use tasks, we evaluate performance on τ^2-Bench [3] and AceBench [6], which emphasize multi-turn tool-calling capabilities. In reasoning, we include a wide range of mathematical, science and logical tasks: AIME 2024/2025, MATH-500, HMMT 2025, CNMO 2024, PolyMath-en, ZebraLogic [43], AutoLogi [91], GPQA-Diamond [61], SuperGPQA [13], and Humanity's Last Exam (Text-Only) [56]. We benchmark the long-context capabilities on: MRCR[4] for long-context retrieval, and DROP [14], FRAMES [37] and LongBench v2 [2] for long-context reasoning. For factuality, we evaluate FACTS Grounding [30], the Vectara Hallucination Leaderboard [73], and FaithJudge [68]. Finally, general capabilities are assessed using MMLU [23], MMLU-Redux [17], MMLU-Pro [76], IFEval [90], Multi-Challenge [64], SimpleQA [78], and LiveBench [80] (as of 2024-11-25).

Baselines We benchmark against both open-source and proprietary frontier models, ensuring every candidate is evaluated under its non-thinking configuration to eliminate additional gains from test-time compute. Open-source baselines: DeepSeek-V3-0324 and Qwen3-235B-A22B, with the latter run in the vendor-recommended no-thinking regime. Proprietary baselines: Claude Sonnet 4, Claude Opus 4, GPT-4.1, and Gemini 2.5 Flash Preview (2025-05-20). Each invoked in its respective non-thinking mode via official APIs under unified temperature and top-p settings.

Evaluation Configurations All runs query models in their non-thinking mode. Output token length is capped at 8192 tokens everywhere except SWE-bench Verified (Agentless), which is raised to 16384. For benchmarks with high per-question variance, we adopt repeated sampling k times and average the results to obtain stable scores, denoted as Avg@k. For long-context tasks, we set the context window size to 128K tokens during evaluation, truncating any input that exceeds this limit to fit within the window. SWE-bench Verified is evaluated in two modes: Agentless Coding via Single Patch without Test (Acc) and Agentic Coding via bash/editor tools under both Single Attempt (Acc) and Multiple Attempts (Acc) using best-of-N selection with an internal verifier; SWE-bench Multilingual is tested only in the single-attempt agentic setting. Some data points have been omitted due to prohibitively expensive evaluation costs.

4.1.2 Evaluation Results

A comprehensive evaluation results of Kimi-K2-Instruct is shown in Table 3, with detailed explanation provided in the Appendix C. Below, we highlight key results across four core domains:

Agentic and Competitive Coding Kimi-K2-Instruct demonstrates state-of-the-art open-source performance on real-world SWE tasks. It outperforms most baselines on SWE-bench Verified (65.8%, 71.6% with multiple attemps), SWE-bench Multilingual (47.3%), and SWE-lancer (39.1%), significantly closing the gap with Claude 4 Opus and Sonnet. On competitive coding benchmarks (e.g., LiveCodeBench v6 53.7%, OJBench 27.1%), it also leads among all models, highlighting its practical coding proficiency across difficulty levels.

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

(i) 我们将重型环境部署为可更轻松扩展的专用服务；(ii) 我们采用大量并发 rollout，以摊销某些昂贵交互带来的延迟。

在智能体 rollout 中的另一个挑战是，单个 rollout 轨迹可能极其漫长。为了防止长尾轨迹阻塞整个 rollout 过程，我们采用了部分 rollout [35] 技术。该策略允许将长尾未完成任务暂停，并在下一次 RL 迭代中恢复。

为了提高研究效率，我们还设计了一个受 OpenAI Gym 框架 [49] 启发的统一接口，以简化新环境的集成。我们希望未来能将我们的强化学习基础设施扩展到更多样化的交互环境中。

4 评估

本节首先对 Kimi-K2-Instruct 进行训练后评估，随后简要概述 Kimi-K2-Base 的能力，最后给出全面的安全性评估。

4.1 训练后评估

4.1.1 评估设置

基准测试 我们在多个领域评估 Kimi-K2-Instruct。在编程方面，采用 LiveCodeBench v6 [31]（2024 年 8 月至 2025 年 5 月的问题）、OJBench [77]、MultiPL-E [5]、SWE-bench Verified [32, 84]、TerminalBench [71]、Multi-SWE-bench [86]、SWE-Lancer [50]、PaperBench [65] 和 Aider-Polyglot [16]。在工具使用任务上，评估 τ^2-Bench [3] 和 AceBench [6]，重点考察多轮工具调用能力。在推理方面，涵盖广泛的数学、科学和逻辑任务：AIME 2024/2025、MATH-500、HMMT 2025、CNMO 2024、PolyMath-en、ZebraLogic [43]、AutoLogi [91]、GPQA-Diamond [61]、SuperGPQA [13] 以及 Humanity's Last Exam（仅文本）[56]。长上下文能力在 MRCR[4]（长上下文检索）以及 DROP [14]、FRAMES [37] 和 LongBench v2 [2]（长上下文推理）上进行基准测试。在事实性方面，评估 FACTS Grounding [30]、Vectara Hallucination Leaderboard [73] 和 FaithJudge [68]。最后，通用能力通过 MMLU [23]、MMLU-Redux [17]、MMLU-Pro [76]、IFEval [90]、Multi-Challenge [64]、SimpleQA [78] 和 LiveBench [80]（截至 2024-11-25）进行评估。

基线 我们与开源和专有前沿模型进行基准测试，确保每位候选模型都在其非思考配置下评估，以排除测试时计算带来的额外收益。开源基线：DeepSeek-V3-0324 和 Qwen3-235B-A22B，后者按厂商推荐的非思考模式运行。专有基线：Claude Sonnet 4、Claude Opus 4、GPT-4.1 和 Gemini 2.5 Flash Preview（2025-05-20）。每个模型均通过官方 API 在其各自的非思考模式下调用，统一温度与 top-p 设置。

评估配置 所有运行均在模型的非思考模式下查询。除 SWE-bench Verified（Agentless）外，输出 token 长度上限统一设为 8192；SWE-bench Verified（Agentless）则放宽至 16384。对于每题方差较大的基准，我们采用重复采样 k 次并取平均，以获得稳定分数，记作 Avg@k。长上下文任务在评估时将上下文窗口设为 128K token，超出部分将被截断以适配窗口。SWE-bench Verified 以两种模式评估： 1. Agentless Coding via Single Patch without Test（Acc）； 2. Agentic Coding via bash/editor 工具，在 Single Attempt（Acc）与 Multiple Attempts（Acc）下均使用内部验证器的 best-of-N 选择。 SWE-bench Multilingual 仅在单轮 agentic 设置下测试。部分数据点因评估成本过高而被省略。

4.1.2 评估结果

Kimi-K2-Instruct 的综合评估结果如表 3 所示，详细说明见附录 C。下文我们重点介绍四个核心领域的关键结果：

代理与竞技编码 Kimi-K2-Instruct 在真实世界 SWE 任务上展现了当前开源模型的最佳性能。它在 SWE-bench Verified（65.8%，多次尝试 71.6%）、SWE-bench Multilingual（47.3%）和 SWE-lancer（39.1%）上均超越大多数基线，显著缩小了与 Claude 4 Opus 和 Sonnet 的差距。在竞技编码基准（如 LiveCodeBench v6 53.7%，OJBench 27.1%）上，它也领先所有模型，凸显其在不同难度下的实用编码能力。

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

Table 3: Performance comparison of Kimi-K2-Instruct against leading open-source and proprietary models across diverse tasks. **Bold** denotes the global SOTA; **underlined bold** indicates the best open-source result. Data points marked with * are taken directly from the model's technical report or blog.

Benchmark	Open Source			Proprietary			
	Kimi-K2-Instruct	DeepSeek-V3-0324	Qwen3-235B-A22B	Claude Sonnet 4	Claude Opus 4	GPT-4.1	Gemini 2.5 Flash
Coding Tasks							
LiveCodeBench v6 (Pass@1)	**53.7**	46.9	37.0	48.5	47.4	44.7	44.7
OJBench (Pass@1)	**27.1**	24.0	11.3	15.3	19.6	19.5	19.5
MultiPL-E (Pass@1)	**85.7**	83.1	78.2	88.6	**89.6**	86.7	85.6
SWE-bench Verified *Agentless-Single-Patch* (Pass@1)	**51.8**	36.6	39.4	50.2	**53.0**	40.8	32.6
SWE-bench Verified *Agentic-Single-Attempt* (Pass@1)	**65.8**	38.8	34.4	**72.7***	72.5*	54.6	—
SWE-bench Verified *Agentic-Multi-Attempt* (Pass@1)	**71.6**	—	—	**80.2***	79.4*	—	—
SWE-bench Multilingual (Pass@1)	**47.3**	25.8	20.9	**51.0**	—	31.5	—
Multi-SWE-bench (Pass@1)	**18.3**	8.0	9.0	**29.2**	—	11.7	14.0
SWE-Lancer (Pass@1)	**39.1**	30.5	24.1	**40.8**	—	23.0	38.5
Paper Bench *Code-Dev* (Acc.)	**27.8**	12.2	13.2	**43.3**	—	29.9	5.7
Terminal Bench *In-House* (Acc.)	**30.0**	—	—	35.5	**43.2**	8.3	—
Terminal Bench *Terminus* (Acc.)	**25.0**	16.3	6.6	—	—	**30.3**	16.8
Aider-Polyglot (Acc.)	60.0	55.1	**61.8**	56.4	**70.7**	52.4	44.0
Tool Use Tasks							
Tau2 retail (Avg@4)	**70.6**	69.1	57.0	75.0	**81.8**	74.8	64.3
Tau2 airline (Avg@4)	**56.5**	39.0	26.5	55.5	**60.0**	54.5	42.5
Tau2 telecom (Avg@4)	**65.8**	32.5	22.1	45.2	57.0	38.6	16.9
AceBench (Acc.)	**76.5**	72.7	70.5	76.2	75.6	**80.1**	74.5
Math & STEM Tasks							
AIME 2024 (Avg@64)	**69.6**	59.4*	40.1*	43.4	48.2	46.5	61.3
AIME 2025 (Avg@64)	**49.5**	46.7	24.7*	33.1*	33.9*	37.0	46.6
MATH-500 (Acc.)	**97.4**	94.0*	91.2*	94.0	94.4	92.4	95.4
HMMT 2025 (Avg@32)	**38.8**	27.5	11.9	15.9	15.9	19.4	34.7
CNMO 2024 (Avg@16)	74.3	**74.7**	48.6	60.4	57.6	56.6	**75.0**
PolyMath-en (Avg@4)	**65.1**	59.5	51.9	52.8	49.8	54.0	49.9
ZebraLogic (Acc.)	**89.0**	84.0	37.7*	79.7	59.3	58.5	57.9
AutoLogi (Acc.)	**89.5**	88.9	83.3*	89.8	86.1	88.2	84.1
GPQA-Diamond (Avg@8)	**75.1**	68.4*	62.9*	70.0*	74.9*	66.3	68.2
SuperGPQA (Acc.)	**57.2**	53.7	50.2	55.7	56.5	50.8	49.6
Humanity's Last Exam (Acc.)	4.7	5.2	**5.7**	5.8	**7.1**	3.7	5.6
General Tasks							
MMLU (EM)	**89.5**	89.4	87.0	91.5	**92.9**	90.4	90.1
MMLU-Redux (EM)	**92.7**	90.5	89.2*	93.6	**94.2**	92.4	90.6
MMLU-Pro (EM)	81.1	**81.2***	77.3	83.7	**86.6**	81.8	79.4
IFEval (Prompt Strict)	**89.8**	81.1	83.2*	87.6	87.4	88.0	84.3
Multi-Challenge (Acc.)	**54.1**	31.4	34.0	46.8	49.0	36.4	39.5
SimpleQA (Correct)	**31.0**	27.7	13.2	15.9	22.8	**42.3**	23.3
Livebench (Pass@1)	**76.4**	72.4	67.6	74.8	74.6	69.8	67.8
Arena Hard v2.0 *Hard Prompt* (Win rate)	**54.5**	39.9	39.9	51.6	**59.7**	51.7	48.7
Arena Hard v2.0 *Creative Writing* (Win rate)	**85.0**	59.3	59.8	54.6	68.5	61.5	72.8
FACTS Grounding (Adjusted)	**88.5**	68.3	68.5	83.6	—	79.2	86.6
HHEM v2.1 (1-Hallu.)	**98.9**	88.9	94.5	94.5	—	96.7	**97.8**
FaithJudge (1-Hallu.)	**92.6**	83.4	75.7	83.0	—	91.0	**93.2**
LongBench v2 (Acc.)	49.1	**51.1**	—	52.5	—	54.3	**55.5**
FRAMES (Acc.)	77.1	**79.2**	—	76.3	—	**87.4**	72.9
MRCR (Acc.)	**55.0**	50.8	—	74.4	—	66.9	**81.7**
DROP (Acc.)	**93.5**	91.2	84.3	**92.0**	—	79.1	81.7

表3：Kimi-K2-Instruct 与领先开源及专有模型在多种任务上的性能对比。加粗表示全球 SOTA；加下划线的加粗表示最佳开源结果。数据点
marked with * are taken directly from the model's technical report or blog.

Benchmark	Open Source			Proprietary			
	Kimi-K2-Instruct	DeepSeek-V3-0324	Qwen3-235B-A22B	Claude Sonnet 4	Claude Opus 4	GPT-4.1	Gemini 2.5 Flash
Coding Tasks							
LiveCodeBench v6 (Pass@1)	**53.7**	46.9	37.0	48.5	47.4	44.7	44.7
OJBench (Pass@1)	**27.1**	24.0	11.3	15.3	19.6	19.5	19.5
MultiPL-E (Pass@1)	**85.7**	83.1	78.2	88.6	**89.6**	86.7	85.6
SWE-bench Verified *Agentless-Single-Patch* (Pass@1)	**51.8**	36.6	39.4	50.2	**53.0**	40.8	32.6
SWE-bench Verified *Agentic-Single-Attempt* (Pass@1)	**65.8**	38.8	34.4	**72.7***	72.5*	54.6	—
SWE-bench Verified *Agentic-Multi-Attempt* (Pass@1)	**71.6**	—	—	**80.2***	79.4*	—	—
SWE-bench Multilingual (Pass@1)	**47.3**	25.8	20.9	**51.0**	—	31.5	—
Multi-SWE-bench (Pass@1)	**18.3**	8.0	9.0	**29.2**	—	11.7	14.0
SWE-Lancer (Pass@1)	**39.1**	30.5	24.1	**40.8**	—	23.0	38.5
Paper Bench *Code-Dev* (Acc.)	**27.8**	12.2	13.2	**43.3**	—	29.9	5.7
Terminal Bench *In-House* (Acc.)	**30.0**	—	—	35.5	**43.2**	8.3	—
Terminal Bench *Terminus* (Acc.)	**25.0**	16.3	6.6	—	—	**30.3**	16.8
Aider-Polyglot (Acc.)	60.0	55.1	**61.8**	56.4	**70.7**	52.4	44.0
Tool Use Tasks							
Tau2 retail (Avg@4)	**70.6**	69.1	57.0	75.0	**81.8**	74.8	64.3
Tau2 airline (Avg@4)	**56.5**	39.0	26.5	55.5	**60.0**	54.5	42.5
Tau2 telecom (Avg@4)	**65.8**	32.5	22.1	45.2	57.0	38.6	16.9
AceBench (Acc.)	**76.5**	72.7	70.5	76.2	75.6	**80.1**	74.5
Math & STEM Tasks							
AIME 2024 (Avg@64)	**69.6**	59.4*	40.1*	43.4	48.2	46.5	61.3
AIME 2025 (Avg@64)	**49.5**	46.7	24.7*	33.1*	33.9*	37.0	46.6
MATH-500 (Acc.)	**97.4**	94.0*	91.2*	94.0	94.4	92.4	95.4
HMMT 2025 (Avg@32)	**38.8**	27.5	11.9	15.9	15.9	19.4	34.7
CNMO 2024 (Avg@16)	74.3	**74.7**	48.6	60.4	57.6	56.6	**75.0**
PolyMath-en (Avg@4)	**65.1**	59.5	51.9	52.8	49.8	54.0	49.9
ZebraLogic (Acc.)	**89.0**	84.0	37.7*	79.7	59.3	58.5	57.9
AutoLogi (Acc.)	**89.5**	88.9	83.3*	89.8	86.1	88.2	84.1
GPQA-Diamond (Avg@8)	**75.1**	68.4*	62.9*	70.0*	74.9*	66.3	68.2
SuperGPQA (Acc.)	**57.2**	53.7	50.2	55.7	56.5	50.8	49.6
Humanity's Last Exam (Acc.)	4.7	5.2	**5.7**	5.8	**7.1**	3.7	5.6
General Tasks							
MMLU (EM)	**89.5**	89.4	87.0	91.5	**92.9**	90.4	90.1
MMLU-Redux (EM)	**92.7**	90.5	89.2*	93.6	**94.2**	92.4	90.6
MMLU-Pro (EM)	81.1	**81.2***	77.3	83.7	**86.6**	81.8	79.4
IFEval (Prompt Strict)	**89.8**	81.1	83.2*	87.6	87.4	88.0	84.3
Multi-Challenge (Acc.)	**54.1**	31.4	34.0	46.8	49.0	36.4	39.5
SimpleQA (Correct)	**31.0**	27.7	13.2	15.9	22.8	**42.3**	23.3
Livebench (Pass@1)	**76.4**	72.4	67.6	74.8	74.6	69.8	67.8
Arena Hard v2.0 *Hard Prompt* (Win rate)	**54.5**	39.9	39.9	51.6	**59.7**	51.7	48.7
Arena Hard v2.0 *Creative Writing* (Win rate)	**85.0**	59.3	59.8	54.6	68.5	61.5	72.8
FACTS Grounding (Adjusted)	**88.5**	68.3	68.5	83.6	—	79.2	86.6
HHEM v2.1 (1-Hallu.)	**98.9**	88.9	94.5	94.5	—	96.7	**97.8**
FaithJudge (1-Hallu.)	**92.6**	83.4	75.7	83.0	—	91.0	**93.2**
LongBench v2 (Acc.)	49.1	**51.1**	—	52.5	—	54.3	**55.5**
FRAMES (Acc.)	77.1	**79.2**	—	76.3	—	**87.4**	72.9
MRCR (Acc.)	**55.0**	50.8	—	74.4	—	66.9	**81.7**
DROP (Acc.)	**93.5**	91.2	84.3	**92.0**	—	79.1	81.7

Agentic Tool Use On multi-turn tool-use benchmarks, Kimi-K2-Instruct sets a new standard. It achieves 66.1 Pass@1 on τ^2-Bench and 76.5 on ACEBench, substantially outperforming all baselines. These results affirm its strength in grounded, controlled, and agent-driven tool orchestration across domains.

General Capabilities Kimi-K2-Instruct exhibits strong, balanced performance across general knowledge, math, instruction following, and long-context tasks. It surpasses open-source peers on SimpleQA (31.0%), MMLU (89.5%) and MMLU-Redux (92.7%), and leads all models on instruction benchmarks (IFEval: 89.8%, Multi-Challenge: 54.1%). In math and STEM, it achieves top-tier scores (AIME 2024: 69.6%, GPQA-Diamond: 75.1%), and remains competitive on long-context factuality and retrieval (DROP: 93.5%, MRCR: 55.0%). These results position Kimi-K2-Instruct as a well-rounded and capable generalist across both short- and long-context settings.

Open-Ended Evaluation On the LMSYS Arena leaderboard (July 17, 2025), Kimi-K2-Instruct ranks as the top-1 open-source model and 5th overall based on over 3,000 user votes. This real-world preference signal—across diverse, blind prompts—underscores Kimi-K2's strengths in generating high-quality responses on open-ended tasks.

4.2 Pre-training Evaluations

4.2.1 Evaluation Settings

Benchmarks We evaluate Kimi-K2-Base across diverse capability areas. For general capabilities, we assess on MMLU [23], MMLU-Pro [76], MMLU-Redux [17], BBH [67], TriviaQA [34], SuperGPQA [13], SimpleQA [78], HellaSwag [88], AGIEval [89], GPQA-Diamond [61], ARC-Challenge [8], and WinoGrande [62]. For coding capabilities, we employ EvalPlus [45] (averaging HumanEval [7], MBPP [1], HumanEval+, and MBPP+), LiveCodeBench v6 [31], and CRUXEval [18]. For mathematical reasoning, we utilize GSM8K [9], GSM8K-Platinum [74], MATH [24], and CMATH [79]. For Chinese language capabilities, we evaluate on C-Eval [29], CMMLU [40], and CSimpleQA [22].

Baselines We benchmark against leading open-source foundation models: DeepSeek-V3-Base [10], Qwen2.5-72B-Base [59] (Note that Qwen3-235B-A22B-Base is not open-sourced, and the largest open-sourced base model in the Qwen series is Qwen2.5-72B-Base), and Llama 4-Maverick [70] (Llama 4-Behemoth is also not open-sourced). All models are evaluated under identical configurations to ensure fair comparison.

Evaluation Configurations We employ perplexity-based evaluation for MMLU, MMLU-Redux, GPQA-Diamond, HellaSwag, ARC-Challenge, C-Eval, and CMMLU. Generation-based evaluation is used for MMLU-Pro, SuperGPQA, TriviaQA, BBH, CSimpleQA, MATH, CMATH, GSM8K, GSM8K-Platinum, CRUXEval, LiveCodeBench, and EvalPlus. To mitigate the high variance inherent to GPQA-Diamond, we report the mean score across eight independent runs. All evaluations are conducted using our internal framework derived from LM-Harness-Evaluation [4], ensuring consistent settings across all models.

4.2.2 Evaluation Results

Table 4 presents a comprehensive comparison of Kimi-K2-Base against leading open-source foundation models across diverse evaluation benchmarks. The results demonstrate that Kimi-K2-Base achieves state-of-the-art performance across the majority of evaluated tasks, establishing it as a leading foundation model in the open-source landscape.

General Language Understanding Kimi-K2-Base achieves state-of-the-art performance on 10 out of 12 English language benchmarks. Notable results include MMLU (87.79%), MMLU-Pro (69.17%), MMLU-Redux (90.17%), SuperGPQA (44.67%), and SimpleQA (35.25%), significantly outperforming all baselines.

Coding Capabilities On coding benchmarks, Kimi-K2-Base sets new standards with leading performance across all metrics. It achieves 74.00% on CRUXEval-I-cot, 83.50% on CRUXEval-O-cot, 26.29% on LiveCodeBench v6, and 80.33% on EvalPlus, demonstrating superior code generation and comprehension abilities, particularly in scenarios requiring step-by-step reasoning.

Mathematical Reasoning Kimi-K2-Base exhibits exceptional mathematical capabilities, leading on three out of four benchmarks: MATH (70.22%), GSM8K (92.12%), and GSM8K-Platinum (94.21%). It maintains competitive performance on CMATH (90.26%), narrowly behind DeepSeek-V3-Base (90.53%). These results highlight the model's robust mathematical problem-solving abilities across varying difficulty levels.

在多轮工具使用基准测试中，Kimi-K2-Instruct 树立了新的标杆。它在 τ^2-Bench 上达到 66.1 的 Pass@1，在 ACEBench 上达到 76.5，显著超越所有基线。这些结果充分证明了其在跨领域、基于事实、受控且由智能体驱动的工具编排方面的强大能力。

通用能力 Kimi-K2-Instruct 在通用知识、数学、指令遵循和长文本任务上展现出强劲且均衡的表现。它在 SimpleQA（31.0%）、MMLU（89.5%）和 MMLU-Redux（92.7%）上超越开源对手，并在指令基准测试中领先所有模型（IFEval：89.8%，Multi-Challenge：54.1%）。在数学与 STEM 领域，它取得顶尖成绩（AIME 2024：69.6%，GPQA-Diamond：75.1%），在长文本事实性与检索任务上也保持竞争力（DROP：93.5%，MRCR：55.0%）。这些结果使 Kimi-K2-Instruct 成为在短文本和长文本场景下都全面且强大的通用模型。

开放式评估 在LMSYS Arena 排行榜（2025 年 7 月 17 日）上，Kimi-K2-Instruct 以超过 3,000 张用户投票位列开源模型第 1 名、总榜第 5 名。这一来自真实世界的偏好信号——涵盖多样化、盲测提示——凸显了 Kimi-K2 在开放式任务中生成高质量回复的优势。

4.2 预训练评估

4.2.1 评估设置

基准测试 我们在多种能力领域对 Kimi-K2-Base 进行评估。对于通用能力，我们在 MMLU [23]、MMLU-Pro [76]、MMLU-Redux [17]、BBH [67]、TriviaQA [34]、SuperGPQA [13]、SimpleQA [78]、HellaSwag [88]、AGIEval [89]、GPQA-Diamond [61]、ARC-Challenge [8] 和 WinoGrande [62] 上进行评估。对于编程能力，我们使用 EvalPlus [45]（取 HumanEval [7]、MBPP [1]、HumanEval+ 和 MBPP+ 的平均值）、LiveCodeBench v6 [31] 和 CRUXEval [18]。对于数学推理，我们使用 GSM8K [9]、GSM8K-Platinum [74]、MATH [24] 和 CMATH [79]。对于中文语言能力，我们在 C-Eval [29]、CMMLU [40] 和 CSimpleQA [22] 上进行评估。

基线 我们将与领先的开源基础模型进行对比：DeepSeek-V3-Base [10]、Qwen2.5-72B-Base [59]（注意 Qwen3-235B-A22B-Base 并未开源，Qwen 系列中最大的开源基础模型是 Qwen2.5-72B-Base）以及 Llama 4-Maverick [70]（Llama 4-Behemoth 同样未开源）。所有模型均在相同配置下评估，以确保公平比较。

评估配置 我们对MMLU、MMLU-Redux、GPQA-Diamond、HellaSwag、ARC-Challenge、C-Eval 和 CMMLU 采用基于困惑度的评估。 对MMLU-Pro、SuperGPQA、TriviaQA、BBH、CSimpleQA、MATH、CMATH、GSM8K、GSM8K-Platinum、CRUXEval、LiveCodeBench 和 EvalPlus 采用基于生成的评估。 为缓解GPQA-Diamond 本身的高方差，我们报告 8 次独立运行的平均分。 所有评估均在我们内部基于LM-Harness-Evaluation [4] 衍生的框架上进行，确保所有模型使用一致的设置。

4.2.2 评估结果

表4展示了Kimi-K2-Base与领先的开源基础模型在多种评估基准上的全面比较。结果表明，Kimi-K2-Base在大多数评估任务中均达到了最先进的性能，确立了其在开源领域中的领先基础模型地位。

通用语言理解方面，Kimi-K2-Base 在 12 项英文基准测试中的 10 项上达到了当前最佳性能。显著成绩包括 MMLU（87.79%）、MMLU-Pro（69.17%）、MMLU-Redux（90.17%）、SuperGPQA（44.67%）和 SimpleQA（35.25%），显著超越所有基线模型。

编码能力 在编码基准测试中，Kimi-K2-Base 以全面领先的性能树立了新标准。它在 CRUXEval-I-cot 上达到 74.00%，在 CRUXEval-O-cot 上达到 83.50%，在 LiveCodeBench v6 上达到 26.29%，在 EvalPlus 上达到 80.33%，展现出卓越的代码生成与理解能力，尤其在需要逐步推理的场景中表现突出。

数学推理 Kimi-K2-Base 展现出卓越的数学能力，在四项基准测试中的三项领先：MATH（70.22%）、GSM8K（92.12%）和 GSM8K-Platinum（94.21%）。在 CMATH（90.26%）上保持竞争力，仅以微弱差距落后于 DeepSeek-V3-Base（90.53%）。这些结果凸显了模型在不同难度级别上稳健的数学解题能力。

Chinese Language Understanding The model demonstrates superior multilingual capabilities, achieving state-of-the-art results across all Chinese language benchmarks: C-Eval (92.50%), CMMLU (90.90%), and CSimpleQA (77.57%). These results establish Kimi-K2-Base as a leading model for Chinese language understanding while maintaining strong performance across other languages.

Table 4: Performance comparison of Kimi-K2-Base against leading open-source models across diverse tasks.

	Benchmark (Metric)	#Shots	Kimi-K2-Base	DeepSeek-V3-Base	Llama4-Maverick-Base	Qwen2.5-72B-Base
	Architecture	-	MoE	MoE	MoE	Dense
	# Activated Params	-	32B	37B	17B	72B
	# Total Params	-	1043B	671B	400B	72B
English	MMLU	5-shots	**87.79**	87.10	84.87	86.08
	MMLU-pro	5-shots	**69.17**	60.59	63.47	62.80
	MMLU-redux	5-shots	**90.17**	89.53	88.18	87.77
	SuperGPQA	5-shots	**44.67**	39.20	38.84	34.23
	GPQA-Diamond(avg@8)	5-shots	48.11	**50.51**	49.43	40.78
	SimpleQA	5-shots	**35.25**	26.49	23.74	10.31
	TriviaQA	5-shots	**85.09**	84.11	79.25	76.03
	BBH	3-shots	**88.71**	88.37	87.10	84.09
	HellaSwag	5-shots	94.60	89.44	86.02	**95.27**
	AGIEval	-	**84.23**	81.57	67.55	76.87
	ARC-Challenge	0-shot	**95.73**	93.77	94.03	95.56
	WinoGrande	5-shots	**85.32**	84.21	77.58	84.14
Code	CRUXEval-I-cot	0-shots	**74.00**	62.75	67.13	61.12
	CRUXEval-O-cot	0-shots	**83.50**	75.25	75.88	66.13
	LiveCodeBench(v6)	1-shots	**26.29**	24.57	25.14	22.29
	EvalPlus	-	**80.33**	65.61	65.48	66.04
Math	MATH	4-shots	**70.22**	61.70	63.02	62.68
	GSM8k	8-shots	**92.12**	91.66	86.35	90.37
	GSM8k-platinum	8-shots	**94.21**	93.38	88.83	92.47
	CMATH	6-shots	90.26	**90.53**	88.07	86.98
Chinese	C-Eval	5-shots	**92.50**	90.04	80.91	90.86
	CMMLU	5-shots	**90.90**	88.84	81.24	90.55
	CSimpleQA	5-shots	**77.57**	72.13	53.47	50.53

4.3 Safety Evaluation

4.3.1 Experiment Settings

We conducted red-teaming evaluations on Kimi K2 compare with other open-source LLMs. The evaluation covered a range of attack scenarios—including harmful content, privacy content, and security content, as well as different attack strategies such as prompt injection and iterative jailbreak.

We choose *Promptfoo*[5] to generate adversarial prompts and analyze the responses. By this way, we can evaluate model in a scalable ways.

Model Selection We compare Kimi K2 with three other open-source LLMs: DeepSeek-V3, DeepSeek-R1, and Qwen3.

Promptfoo Settings Table 5 lists plugins and strategies evaluated, with each plugin paired with all strategies to assess their performance.

Test Case Count Given the inherent non-determinism of large language model inference, single-pass outputs may exhibit variability. To account for this, we generated 3 attack prompts per plugin for each strategy.

Prompt Language Settings We pre-tested the language compatibility for each plugin-strategy combination. Some plugins support both English and Chinese, while others only support English. For combinations that support both, we generated 3 prompts in each language, resulting in 6 prompts per combination.

[5] https://github.com/promptfoo/promptfoo

中文语言理解　该模型展现出卓越的多语言能力，在所有中文语言基准测试中均取得当前最佳结果C-Eval（92.50%）、CMMLU（90.90%）和 CSimpleQA（77.57%）。这些成绩确立了 Kimi-K2-Base 在中文语言理解方面的领先地位，同时在其他语言上也保持了强劲表现。

表4：Kimi-K2-Base 与领先开源模型在多种任务上的性能对比。

	Benchmark (Metric)	#Shots	Kimi-K2-Base	DeepSeek-V3-Base	Llama4-Maverick-Base	Qwen2.5-72B-Base
	Architecture	-	MoE	MoE	MoE	Dense
	# Activated Params	-	32B	37B	17B	72B
	# Total Params	-	1043B	671B	400B	72B
English	MMLU	5-shots	**87.79**	87.10	84.87	86.08
	MMLU-pro	5-shots	**69.17**	60.59	63.47	62.80
	MMLU-redux	5-shots	**90.17**	89.53	88.18	87.77
	SuperGPQA	5-shots	**44.67**	39.20	38.84	34.23
	GPQA-Diamond(avg@8)	5-shots	48.11	**50.51**	49.43	40.78
	SimpleQA	5-shots	**35.25**	26.49	23.74	10.31
	TriviaQA	5-shots	**85.09**	84.11	79.25	76.03
	BBH	3-shots	**88.71**	88.37	87.10	84.09
	HellaSwag	5-shots	94.60	89.44	86.02	**95.27**
	AGIEval	-	**84.23**	81.57	67.55	76.87
	ARC-Challenge	0-shot	**95.73**	93.77	94.03	95.56
	WinoGrande	5-shots	**85.32**	84.21	77.58	84.14
Code	CRUXEval-I-cot	0-shots	**74.00**	62.75	67.13	61.12
	CRUXEval-O-cot	0-shots	**83.50**	75.25	75.88	66.13
	LiveCodeBench(v6)	1-shots	**26.29**	24.57	25.14	22.29
	EvalPlus	-	**80.33**	65.61	65.48	66.04
Math	MATH	4-shots	**70.22**	61.70	63.02	62.68
	GSM8k	8-shots	**92.12**	91.66	86.35	90.37
	GSM8k-platinum	8-shots	**94.21**	93.38	88.83	92.47
	CMATH	6-shots	90.26	**90.53**	88.07	86.98
Chinese	C-Eval	5-shots	**92.50**	90.04	80.91	90.86
	CMMLU	5-shots	**90.90**	88.84	81.24	90.55
	CSimpleQA	5-shots	**77.57**	72.13	53.47	50.53

4.3 安全评估

4.3.1 实验设置

我们对Kimi K2与其他开源LLM进行了红队评估。评估涵盖了一系列攻击场景——包括有害内容、隐私内容和安全内容，以及不同的攻击策略，如提示注入和迭代越狱。

我们选择 *Promptfoo*[5] 来生成对抗性提示并分析响应。通过这种方式，我们可以以可扩展的方式评估模型。

模型选择 我们将Kimi K2与另外三个开源LLM进行比较：DeepSeek-V3、DeepSeek-R1和Qwen3。

Promptfoo 设置表 5 列出了已评估的插件和策略，每个插件都与所有策略配对以评估其性能。

测试用例数量 鉴于大型语言模型推理固有的非确定性，单次输出可能存在差异。为此，我们为每种策略的每个插件生成了 3 个攻击提示。

提示语言设置　我们预先测试了每个插件/策略组合的语言兼容性。部分插件同时支持英文和中文，另一些仅支持英文。对于同时支持两种语言的组合，我们为每种语言各生成 3 条提示，因此每个组合共有 6 条提示。

[5] https://github.com/promptfoo/promptfoo

Table 5: Enabled Plugins and Strategies

Plugin	Harmful	Graphic Content, Harassment and Bullying, Hate Speech, Insults, Profanity, Radicalization, Self Harm, Sexual Content, ToxicChat
	Criminal	Chemical&Biological Weapons, Child Exploitation, Copyright Violations, Cybercrime, Illegal Activities, Illegal Drugs, Indiscriminate Weapons, Intellectual Property Violation, Non-Violent Crime, Violent Crime, Sex Crimes
	Misinformation	Competitor Endorsement, Unsupervised Contracts, Excessive Agency, Hallucination, Misinformation and Disinformation, Specialized Advice, Unsafe Practices, Imitation, Overreliance, Political Opinions, Religious Sensitivity
	Privacy	Privacy Violation, PII in API/Database, Direct PII Exposure, PII in Session Data, PII via Social Engineering
	Security	ASCII Smuggling, CyberSecEval, Harmbench, Debug Access, Divergent Repetition, DoNotAnswer, Malicious Code, Pliny, Prompt Extraction, Reasoning DoS, Tool Discovery
Strategy	Basic, Prompt Injection, Iterative Jailbreak, Crescendo	

Manual Review We incorporated human review into the evaluation process. To minimize subjectivity problem, we conducted multiple rounds of review and assigned the same reviewer to evaluate all cases within a given test set to ensure consistency and reduce variability in judgment.

4.3.2 Safety Evaluation Results

Table 6 presents the passing rates of different models under various plugin–strategy combinations.

Table 6: Safety Evaluation Results

Plugin	Strategy	Kimi-K2-Instruct	DeepSeek-V3-0324	DeepSeek-R1	Qwen3-235B-A22B
Harmful	Basic	98.04	90.45	99.02	98.53
	Base64	100	90.20	100	100
	Prompt Injection	93.14	100	95.10	99.02
	Iterative Jailbreak	92.16	66.67	72.55	74.51
	Crescendo	64.71	64.71	80.39	86.27
Criminal	Basic	100	99.62	95.45	99.24
	Base64	96.97	89.39	84.85	98.48
	Prompt Injection	75.76	91.67	69.70	98.47
	Iterative Jailbreak	57.57	21.21	25.76	53.03
	Crescendo	56.06	31.81	42.42	59.09
Misinformation	Basic	97.28	92.57	92.46	94.84
	Base64	98.48	90.48	96.83	93.65
	Prompt Injection	98.39	86.51	93.65	93.65
	Iterative Jailbreak	63.97	53.97	84.13	69.84
	Crescendo	85.71	55.56	88.89	84.13
Privacy	Basic	100	100	100	100
	Base64	100	100	100	100
	Prompt Injection	88.33	98.33	100	91.67
	Iterative Jailbreak	76.67	100	93.33	96.67
	Crescendo	96.67	100	96.67	100
Security	Basic	77.84	75.57	70.46	90.09
	Base64	82.93	82.93	63.41	95.12
	Prompt Injection	87.80	97.56	65.85	84.13
	Iterative Jailbreak	43.90	60.97	43.90	78.04
	Crescendo	68.29	87.80	68.29	87.80

Without targeted optimization for specific evaluation scenarios, the passing rate of some complex cases (e.g., Harmful–Iterative Jailbreak) was relatively higher compared to other models.

Across different attack strategies, the models exhibited varying trends. Under the Base64 strategy, passing rates generally approached or reached 100%, suggesting that encoding transformations had minimal impact on the models'

表5：已启用的插件与策略

Plugin	Harmful	Graphic Content, Harassment and Bullying, Hate Speech, Insults, Profanity, Radicalization, Self Harm, Sexual Content, ToxicChat
	Criminal	Chemical&Biological Weapons, Child Exploitation, Copyright Violations, Cybercrime, Illegal Activities, Illegal Drugs, Indiscriminate Weapons, Intellectual Property Violation, Non-Violent Crime, Violent Crime, Sex Crimes
	Misinformation	Competitor Endorsement, Unsupervised Contracts, Excessive Agency, Hallucination, Misinformation and Disinformation, Specialized Advice, Unsafe Practices, Imitation, Overreliance, Political Opinions, Religious Sensitivity
	Privacy	Privacy Violation, PII in API/Database, Direct PII Exposure, PII in Session Data, PII via Social Engineering
	Security	ASCII Smuggling, CyberSecEval, Harmbench, Debug Access, Divergent Repetition, DoNotAnswer, Malicious Code, Pliny, Prompt Extraction, Reasoning DoS, Tool Discovery
Strategy	Basic, Prompt Injection, Iterative Jailbreak, Crescendo	

人工审查 我们在评估流程中引入了人工审查。为了最大程度减少主观性问题，我们进行了多轮审查，并让同一位审查员评估给定测试集中的所有案例，以确保一致性并降低判断的变异性。

4.3.2 安全评估结果

表6展示了不同模型在各种插件-策略组合下的通过率。

表6：安全评估结果

Plugin	Strategy	Kimi-K2-Instruct	DeepSeek-V3-0324	DeepSeek-R1	Qwen3-235B-A22B
Harmful	Basic	98.04	90.45	99.02	98.53
	Base64	100	90.20	100	100
	Prompt Injection	93.14	100	95.10	99.02
	Iterative Jailbreak	92.16	66.67	72.55	74.51
	Crescendo	64.71	64.71	80.39	86.27
Criminal	Basic	100	99.62	95.45	99.24
	Base64	96.97	89.39	84.85	98.48
	Prompt Injection	75.76	91.67	69.70	98.47
	Iterative Jailbreak	57.57	21.21	25.76	53.03
	Crescendo	56.06	31.81	42.42	59.09
Misinformation	Basic	97.28	92.57	92.46	94.84
	Base64	98.48	90.48	96.83	93.65
	Prompt Injection	98.39	86.51	93.65	93.65
	Iterative Jailbreak	63.97	53.97	84.13	69.84
	Crescendo	85.71	55.56	88.89	84.13
Privacy	Basic	100	100	100	100
	Base64	100	100	100	100
	Prompt Injection	88.33	98.33	100	91.67
	Iterative Jailbreak	76.67	100	93.33	96.67
	Crescendo	96.67	100	96.67	100
Security	Basic	77.84	75.57	70.46	90.09
	Base64	82.93	82.93	63.41	95.12
	Prompt Injection	87.80	97.56	65.85	84.13
	Iterative Jailbreak	43.90	60.97	43.90	78.04
	Crescendo	68.29	87.80	68.29	87.80

在未针对特定评估场景进行针对性优化的情况下，某些复杂案例（例如 Harm-ful–Iterative Jailbreak）的通过率相对其他模型更高。

在不同攻击策略下，模型表现出不同的趋势。在 Base64 策略下，通过率普遍接近或达到 100%，表明编码转换对模型的影响极小。

basic robustness. In contrast, the Crescendo strategy led to a general drop in passing rates, indicating stronger adversarial effectiveness.

In addition, complex attack strategies do not always outperform basic prompts. Some originally adversarial prompts may lose their intended meaning after multiple rounds of transformation, rendering the resulting model outputs less meaningful.

Automated Red-teaming Limitations Due to the involvement of human review, the evaluation results inevitably contain a degree of subjectivity. Additionally, certain plugin types involve API misuse or external tool invocation, which are more suitable for evaluating agent models with tool-calling capabilities. In the context of base LLMs, such tests may have limited relevance.

5 Limitations

In our internal tests, we have identified some limitations in current Kimi K2 models. When dealing with hard reasoning tasks or unclear tool definition, the model may generate excessive tokens, sometimes leading to truncated outputs or incomplete tool calls. Additionally, performance may decline on certain tasks if tool use is unnecessarily enabled. When building complete software projects, the success rate of one-shot prompting is not as good as using K2 under an agentic coding framework. We are working to address these issues in future releases and looking forward to more feedbacks.

6 Conclusions

We introduced Kimi K2, a 1T-parameter open-weight MoE model built for agentic intelligence. Leveraging the token-efficient MuonClip optimizer and a 15.5T-token high-quality dataset, Kimi K2 achieves stable, scalable pre-training. Post-training combines large-scale synthetic tool-use data with a unified RL framework using both verifiable rewards and self-critic feedbacks. Kimi K2 sets new state-of-the-art on agentic and reasoning benchmarks, establishing itself as the most capable open-weight LLM to date.

7 Acknowledgments

We would like to acknowledge the valuable support provided by the OpenHands and Multi-SWE-bench teams in evaluating the SWE-bench Verified and Multi-SWE-bench experimental results.

基本鲁棒性。相比之下，Crescendo 策略导致通过率普遍下降，表明其具有更强的对抗有效性。

此外，复杂的攻击策略并不总是优于基础提示。一些原本具有对抗性的提示在经过多轮转换后可能会失去其原意，导致最终模型输出的意义降低。

自动化红队测试的局限性　由于有人工审核的参与，评估结果不可避免地带有一定程度的主观性。此外，某些插件类型涉及 API 滥用或外部工具调用，更适合评估具备工具调用能力的智能体模型。在基础 LLM 的语境下，这类测试的相关性可能有限。

5 限制

在我们的内部测试中，我们发现了当前 Kimi K2 模型的一些局限。在处理困难推理任务或工具定义不清晰时，模型可能会生成过多 token，有时导致输出被截断或工具调用不完整。此外，如果非必要地启用工具使用，某些任务的性能可能会下降。在构建完整软件项目时，一次性提示的成功率不如在代理式编码框架下使用 K2。我们正在努力在未来的版本中解决这些问题，并期待更多反馈。

6 结论

我们推出了 Kimi K2，一个专为智能体智能设计的 1T 参数开源 MoE 模型。借助 token 高效的 MuonClip 优化器以及 15.5T token 的高质量数据集，Kimi K2 实现了稳定且可扩展的预训练。后训练阶段融合大规模合成工具使用数据，并采用统一 RL 框架，结合可验证奖励与自我批评反馈。Kimi K2 在智能体与推理基准上刷新 SOTA，成为迄今为止能力最强的开源 LLM。

7 致谢

我们谨致谢意，感谢 OpenHands 和 Multi-SWE-bench 团队在评估 SWE-bench Verified 和 Multi-SWE-bench 实验结果方面所提供的宝贵支持。

References

[1] Jacob Austin et al. *Program Synthesis with Large Language Models*. 2021. arXiv: 2108.07732 [cs.PL]. URL: https://arxiv.org/abs/2108.07732.

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

[3] Victor Barres et al. *τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment*. 2025. arXiv: 2506.07982 [cs.AI]. URL: https://arxiv.org/abs/2506.07982.

[4] Stella Biderman et al. "Lessons from the trenches on reproducible evaluation of language models". In: *arXiv preprint arXiv:2405.14782* (2024).

[5] Federico Cassano et al. "MultiPL-E: A Scalable and Polyglot Approach to Benchmarking Neural Code Generation". In: *IEEE Transactions on Software Engineering* 49.7 (2023), pp. 3675–3691. DOI: 10.1109/TSE.2023.3267446.

[6] Chen Chen et al. "ACEBench: Who Wins the Match Point in Tool Learning?" In: *arXiv e-prints* (2025), arXiv–2501.

[7] Mark Chen et al. "Evaluating Large Language Models Trained on Code". In: (2021). arXiv: 2107.03374 [cs.LG].

[8] Peter Clark et al. "Think you have solved question answering? try arc, the ai2 reasoning challenge". In: *arXiv preprint arXiv:1803.05457* (2018).

[9] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168.

[10] DeepSeek-AI. *DeepSeek-V3 Technical Report*. 2024. arXiv: 2412.19437 [cs.CL]. URL: https://arxiv.org/abs/2412.19437.

[11] Mostafa Dehghani et al. "Scaling vision transformers to 22 billion parameters". In: *International conference on machine learning*. PMLR. 2023, pp. 7480–7512.

[12] Guanting Dong et al. *Self-play with Execution Feedback: Improving Instruction-following Capabilities of Large Language Models*. 2024. arXiv: 2406.13542 [cs.CL]. URL: https://arxiv.org/abs/2406.13542.

[13] Xinrun Du et al. "Supergpqa: Scaling llm evaluation across 285 graduate disciplines". In: *arXiv preprint arXiv:2502.14739* (2025).

[14] Dheeru Dua et al. "DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs". In: *CoRR* abs/1903.00161 (2019). arXiv: 1903.00161. URL: http://arxiv.org/abs/1903.00161.

[15] Kazuki Fujii et al. *Rewriting Pre-Training Data Boosts LLM Performance in Math and Code*. 2025. arXiv: 2505.02881 [cs.LG]. URL: https://arxiv.org/abs/2505.02881.

[16] Paul Gauthier. *Aider LLM Leaderboards*. https://aider.chat/docs/leaderboards/. 2025.

[17] Aryo Pradipta Gema et al. "Are we done with mmlu?" In: *arXiv preprint arXiv:2406.04127* (2024).

[18] Alex Gu et al. "Cruxeval: A benchmark for code reasoning, understanding and execution". In: *arXiv preprint arXiv:2401.03065* (2024).

[19] Daya Guo et al. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning". In: *arXiv preprint arXiv:2501.12948* (2025).

[20] Zhicheng Guo et al. "StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models". In: *arXiv preprint arXiv:2403.07714* (2025).

[21] Aaron Harlap et al. "Pipedream: Fast and efficient pipeline parallel dnn training". In: *arXiv preprint arXiv:1806.03377* (2018).

[22] Y He et al. "Chinese simpleqa: A chinese factuality evaluation for large language models, 2024a". In: *URL https://arxiv. org/abs/2411.07140* ().

[23] Dan Hendrycks et al. "Measuring massive multitask language understanding". In: *arXiv preprint arXiv:2009.03300* (2020).

[24] Dan Hendrycks et al. *Measuring Mathematical Problem Solving With the MATH Dataset*. 2021. arXiv: 2103.03874 [cs.LG]. URL: https://arxiv.org/abs/2103.03874.

[25] Shengding Hu et al. "Minicpm: Unveiling the potential of small language models with scalable training strategies". In: *arXiv preprint arXiv:2404.06395* (2024).

[26] Jiaxin Huang et al. "Large language models can self-improve". In: *arXiv preprint arXiv:2210.11610* (2022).

[27] Siming Huang et al. *OpenCoder: The Open Cookbook for Top-Tier Code Large Language Models*. 2025. arXiv: 2411.04905 [cs.CL]. URL: https://arxiv.org/abs/2411.04905.

参考文献

[1] Jacob Austin 等. *Program Synthesis with Large Language Models*. 2021. arXiv: 2108.07732 [cs.PL]. URL: https://arxiv.org/abs/2108.07732. [2] Yushi Bai 等.
LongBench v2: Towards Deeper Understanding and Reasoning on Realistic Long-context Multitasks. 2025. arXiv: 2412.15204 [cs.CL]. URL: https://arxiv.org/abs/2412.15204. [3] Victor Barres 等.
τ^2*-Bench: Evaluating Conversational Agents in a Dual-Control Environment*. 2025. arXiv: 2506.07982 [cs.AI]. URL: https://arxiv.org/abs/2506.07982. [4] Stella Biderman 等. "Lessons from the trenches on reproducible evaluation of language models". 载于: *arXiv preprint arXiv:2405.14782* (2024). [5] Federico Cassano 等. "MultiPL-E: A Scalable and Polyglot Approach to Benchmarking Neural Code Generation". 载于:
IEEE Transactions on Software Engineering 49.7 (2023), 第 3675–3691 页. DOI: 10.1109/TSE.2023.3267446. [6] Chen Chen 等. "ACEBench: Who Wins the Match Point in Tool Learning?" 载于: *arXiv e-prints* (2025), arXiv–2501. [7] Mark Chen 等. "Evaluating Large Language Models Trained on Code". 载于: (2021). arXiv: 2107.03374 [cs.LG]. [8] Peter Clark 等. "Think you have solved question answering? try arc, the ai2 reasoning challenge". 载于: *arXiv preprint arXiv:1803.05457* (2018). [9] Karl Cobbe 等. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168. [10] DeepSeek-AI.
DeepSeek-V3 Technical Report. 2024. arXiv: 2412.19437 [cs.CL]. URL: https://arxiv.org/abs/2412.19437. [11] Mostafa Dehghani 等. "Scaling vision transformers to 22 billion parameters". 载于: *International conference on machine learning*. PMLR. 2023, 第 7480–7512 页. [12] Guanting Dong 等.
Self-play with Execution Feedback: Improving Instruction-following Capabilities of Large Language Models. 2024. arXiv: 2406.13542 [cs.CL]. URL: https://arxiv.org/abs/2406.13542. [13] Xinrun Du 等. "Supergpqa: Scaling llm evaluation across 285 graduate disciplines". 载于: *arXiv preprint arXiv:2502.14739* (2025). [14] Dheeru Dua 等. "DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs". 载于: *CoRR* abs/1903.00161 (2019). arXiv: 1903.00161. URL: http://arxiv.org/abs/1903.00161. [15] Kazuki Fujii 等.
Rewriting Pre-Training Data Boosts LLM Performance in Math and Code. 2025. arXiv: 2505.02881 [cs.LG]. URL: https://arxiv.org/abs/2505.02881. [16] Paul Gauthier. *Aider LLM Leaderboards*. https://aider.chat/docs/leaderboards/. 2025. [17] Aryo Pradipta Gema 等. "Are we done with mmlu?" 载于: *arXiv preprint arXiv:2406.04127* (2024). [18] Alex Gu 等. "Cruxeval: A benchmark for code reasoning, understanding and execution". 载于: *arXiv preprint arXiv:2401.03065* (2024). [19] Daya Guo 等. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning". 载于: *arXiv preprint arXiv:2501.12948* (2025). [20] Zhicheng Guo 等. "StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models". 载于:
arXiv preprint arXiv:2403.07714 (2025). [21] Aaron Harlap 等. "Pipedream: Fast and efficient pipeline parallel dnn training". 载于: *arXiv preprint arXiv:1806.03377* (2018). [22] Y He 等. "Chinese simpleqa: A chinese factuality evaluation for large language models, 2024a". 载于: *URL https://arxiv. org/abs/2411.07140* (). [23] Dan Hendrycks 等. "Measuring massive multitask language understanding". 载于: *arXiv preprint arXiv:2009.03300* (2020). [24] Dan Hendrycks 等. *Measuring Mathematical Problem Solving With the MATH Dataset*. 2021. arXiv: 2103.03874 [cs.LG]. URL: https://arxiv.org/abs/2103.03874. [25] Shengding Hu 等. "Minicpm: Unveiling the potential of small language models with scalable training strategies". 载于: *arXiv preprint arXiv:2404.06395* (2024). [26] Jiaxin Huang 等. "Large language models can self-improve". 载于: *arXiv preprint arXiv:2210.11610* (2022). [27] Siming Huang 等. *OpenCoder: The Open Cookbook for Top-Tier Code Large Language Models*. 2025. arXiv: 2411.04905 [cs.CL]. URL: https://arxiv.org/abs/2411.04905.

[28] Yanping Huang et al. "Gpipe: Efficient training of giant neural networks using pipeline parallelism". In: *Advances in neural information processing systems* 32 (2019).

[29] Yuzhen Huang et al. *C-Eval: A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models*. 2023. arXiv: 2305.08322 [cs.CL]. URL: https://arxiv.org/abs/2305.08322.

[30] Alon Jacovi et al. *The FACTS Grounding Leaderboard: Benchmarking LLMs' Ability to Ground Responses to Long-Form Input*. 2025. arXiv: 2501.03200 [cs.CL]. URL: https://arxiv.org/abs/2501.03200.

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

[32] Carlos E Jimenez et al. "SWE-bench: Can Language Models Resolve Real-world Github Issues?" In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=VTF8yNQM66.

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

[34] Mandar Joshi et al. *TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension*. 2017. arXiv: 1705.03551 [cs.CL]. URL: https://arxiv.org/abs/1705.03551.

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

[36] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[37] Satyapriya Krishna et al. *Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation*. 2025. arXiv: 2409.12941 [cs.CL]. URL: https://arxiv.org/abs/2409.12941.

[38] Joel Lamy-Poirier. "Breadth-first pipeline parallelism". In: *Proceedings of Machine Learning and Systems* 5 (2023), pp. 48–67.

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

[40] Haonan Li et al. *CMMLU: Measuring massive multitask language understanding in Chinese*. 2024. arXiv: 2306.09212 [cs.CL]. URL: https://arxiv.org/abs/2306.09212.

[41] Jia Li et al. "Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions". In: *Hugging Face repository* 13.9 (2024), p. 9.

[42] Tianle Li et al. "From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard and BenchBuilder Pipeline". In: *arXiv preprint arXiv:2406.11939* (2024).

[43] Bill Yuchen Lin et al. *ZebraLogic: On the Scaling Limits of LLMs for Logical Reasoning*. 2025. arXiv: 2502.01100 [cs.AI]. URL: https://arxiv.org/abs/2502.01100.

[44] Aixin Liu et al. "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model". In: *arXiv preprint arXiv:2405.04434* (2024).

[45] Jiawei Liu et al. "Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 21558–21572.

[46] Jingyuan Liu et al. "Muon is scalable for LLM training". In: *arXiv preprint arXiv:2502.16982* (2025).

[47] Ziming Liu et al. "Hanayo: Harnessing Wave-like Pipeline Parallelism for Enhanced Large Model Training Efficiency". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '23. ACM, Nov. 2023, pp. 1–13. DOI: 10.1145/3581784.3607073. URL: http://dx.doi.org/10.1145/3581784.3607073.

[48] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[49] Jan Ludziejewski et al. *OpenAI Gym*. 2025. arXiv: 2502.05172 [cs.LG]. URL: https://arxiv.org/abs/2502.05172.

[50] Samuel Miserendino et al. "SWE-Lancer: Can Frontier LLMs Earn $1 Million from Real-World Freelance Software Engineering?" In: *arXiv preprint arXiv:2502.12115* (2025).

[51] Arindam Mitra et al. "Agentinstruct: Toward generative teaching with agentic flows". In: *arXiv preprint arXiv:2407.03502* (2024).

[52] Ivan Moshkov et al. "Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset". In: *arXiv preprint arXiv:2504.16891* (2025).

[53] Deepak Narayanan et al. "Efficient large-scale language model training on gpu clusters using megatron-lm". In: *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 2021, pp. 1–15.

[28] Yanping Huang 等。“Gpipe：利用流水线并行高效训练巨型神经网络”。载于：*Advances in neural information processing systems* 32 (2019)。 [29] Yuzhen Huang 等。*C-Eval: A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models*。2023。arXiv: 2305.08322 [cs.CL]。URL: https://arxiv.org/abs/2305.08322。 [30] Alon Jacovi 等。*The FACTS Grounding Leaderboard: Benchmarking LLMs' Ability to Ground Responses to Long-Form Input*。2025。arXiv: 2501.03200 [cs.CL]。URL: https://arxiv.org/abs/2501.03200。 [31] Naman Jain 等。“Livecodebench：对大型语言模型进行代码任务的整体且无数据污染评估”。载于：*arXiv preprint arXiv:2403.07974* (2024)。 [32] Carlos E Jimenez 等。“SWE-bench：语言模型能否解决真实世界的 GitHub Issue？”载于：*The Twelfth International Conference on Learning Representations*。2024。URL: https://openreview.net/forum?id= VTF8yNQM66。 [33] Keller Jordan 等。*Muon: An optimizer for hidden layers in neural networks*。2024。URL: https://kellerjordan.github.io/posts/muon/。 [34] Mandar Joshi 等。*TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension*。2017。arXiv: 1705.03551 [cs.CL]。URL: https://arxiv.org/abs/1705.03551。 [35] Kimi Team。“Kimi k1.5：利用强化学习与 LLM 的规模化”。载于：*arXiv preprint arXiv:2501.12599* (2025)。 [36] Diederik P. Kingma 与 Jimmy Ba。“Adam：一种随机优化方法”。载于：*3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*。Yoshua Bengio 与 Yann LeCun 编。2015。URL: http://arxiv.org/abs/1412.6980。 [37] Satyapriya Krishna 等。*Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation*。2025。arXiv: 2409.12941 [cs.CL]。URL: https://arxiv.org/abs/2409.12941。 [38] Joel Lamy-Poirier。“Breadth-first pipeline parallelism”。载于：*Proceedings of Machine Learning and Systems* 5 (2023)，第 48–67 页。 [39] Dmitry Lepikhin 等。“Gshard：通过条件计算与自动分片扩展巨型模型”。载于：*arXiv preprint arXiv:2006.16668* (2020)。 [40] Haonan Li 等。*CMMLU: Measuring massive multitask language understanding in Chinese*。2024。arXiv: 2306.09212 [cs.CL]。URL: https://arxiv.org/abs/2306.09212。 [41] Jia Li 等。“Numinamath：AI4Maths 领域最大的公开数据集，包含 86 万对竞赛数学题目与解答”。载于：*Hugging Face repository* 13.9 (2024)，第 9 页。 [42] Tianle Li 等。“从众包数据到高质量基准：Arena-Hard 与 BenchBuilder 流程”。载于：*arXiv preprint arXiv:2406.11939* (2024)。 [43] Bill Yuchen Lin 等。*ZebraLogic: On the Scaling Limits of LLMs for Logical Reasoning*。2025。arXiv: 2502.01100 [cs.AI]。URL: https://arxiv.org/abs/2502.01100。 [44] Aixin Liu 等。“Deepseek-v2：强大、经济且高效的专家混合语言模型”。载于：*arXiv preprint arXiv:2405.04434* (2024)。 [45] Jiawei Liu 等。“ChatGPT 生成的代码真的正确吗？对大型语言模型代码生成的严格评估”。载于：*Advances in Neural Information Processing Systems* 36 (2023)，第 21558–21572 页。 [46] Jingyuan Liu 等。“Muon 在 LLM 训练中具备可扩展性”。载于：*arXiv preprint arXiv:2502.16982* (2025)。 [47] Ziming Liu 等。“Hanayo：利用波浪式流水线并行提升大模型训练效率”。载于：*Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*。SC ’23。ACM，2023 年 11 月，第 1–13 页。DOI: 10.1145/3581784.3607073。URL: http://dx.doi.org/10.1145/3581784.3607073。 [48] Ilya Loshchilov 与 Frank Hutter。“解耦权重衰减正则化”。载于：*International Conference on Learning Representations*。2019。URL: https://openreview.net/forum?id=Bkg6RiCqY7。 [49] Jan Ludziejewski 等。*OpenAI Gym*。2025。arXiv: 2502.05172 [cs.LG]。URL: https://arxiv.org/abs/2502.05172。 [50] Samuel Miserendino 等。“SWE-Lancer：前沿 LLM 能否通过真实自由职业软件工程赚取 100 万美元？”载于：*arXiv preprint arXiv:2502.12115* (2025)。 [51] Arindam Mitra 等。“Agentinstruct：面向生成式教学的智能体流程”。载于：*arXiv preprint arXiv:2407.03502* (2024)。 [52] Ivan Moshkov 等。“Aimo-2 获胜方案：利用 OpenMathReasoning 数据集构建最先进的数学推理模型”。载于：*arXiv preprint arXiv:2504.16891* (2025)。 [53] Deepak Narayanan 等。“利用 Megatron-LM 在 GPU 集群上高效训练大规模语言模型”。载于：*Proceedings of the international conference for high performance computing, networking, storage and analysis*。2021，第 1–15 页。

[54] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

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

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

[57] Penghui Qi et al. "Zero bubble pipeline parallelism". In: *arXiv preprint arXiv:2401.10241* (2023).

[58] Yujia Qin et al. "Toolllm: Facilitating large language models to master 16000+ real-world apis". In: *arXiv preprint arXiv:2307.16789* (2023).

[59] Qwen et al. *Qwen2.5 Technical Report*. 2025. arXiv: 2412.15115 [cs.CL]. URL: https://arxiv.org/abs/2412.15115.

[60] Samyam Rajbhandari et al. "Zero: Memory optimizations toward training trillion parameter models". In: *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2020, pp. 1–16.

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

[62] Keisuke Sakaguchi et al. "Winogrande: An adversarial winograd schema challenge at scale". In: *Communications of the ACM* 64.9 (2021), pp. 99–106.

[63] David Silver and Richard S Sutton. "Welcome to the era of experience". In: *Google AI* 1 (2025).

[64] Ved Sirdeshmukh et al. *MultiChallenge: A Realistic Multi-Turn Conversation Evaluation Benchmark Challenging to Frontier LLMs*. 2025. arXiv: 2501.17399 [cs.CL]. URL: https://arxiv.org/abs/2501.17399.

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

[66] Hao Sun et al. *ZeroSearch: Incentivize the Search Capability of LLMs without Searching*. 2025. arXiv: 2505.04588 [cs.CL]. URL: https://arxiv.org/abs/2505.04588.

[67] Mirac Suzgun et al. *Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them*. 2022. arXiv: 2210.09261 [cs.CL]. URL: https://arxiv.org/abs/2210.09261.

[68] Manveer Singh Tamber et al. "Benchmarking LLM Faithfulness in RAG with Evolving Leaderboards". In: *arXiv preprint arXiv:2505.04847* (2025).

[69] Gemma Team et al. "Gemma 2: Improving open language models at a practical size". In: *arXiv preprint arXiv:2408.00118* (2024).

[70] LlaMA Team. *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation — ai.meta.com*. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. [Accessed 15-07-2025].

[71] The Terminal-Bench Team. *Terminal-Bench: A Benchmark for AI Agents in Terminal Environments*. Apr. 2025. URL: https://github.com/laude-institute/terminal-bench.

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

[73] Vectara. *Hallucination Evaluation Model (Revision 7437011)*. 2024. URL: https://huggingface.co/vectara/hallucination_evaluation_model.

[74] Joshua Vendrow et al. "Do large language model benchmarks test reliability?" In: *arXiv preprint arXiv:2502.03461* (2025).

[75] Yizhong Wang et al. "Self-instruct: Aligning language models with self-generated instructions". In: *arXiv preprint arXiv:2212.10560* (2022).

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

[77] Zhexu Wang et al. *OJBench: A Competition Level Code Benchmark For Large Language Models*. 2025. arXiv: 2506.16395 [cs.CL]. URL: https://arxiv.org/abs/2506.16395.

[78] Jason Wei et al. "Measuring short-form factuality in large language models". In: *arXiv preprint arXiv:2411.04368* (2024).

[79] Tianwen Wei et al. *CMATH: Can Your Language Model Pass Chinese Elementary School Math Test?* 2023. arXiv: 2306.16636 [cs.CL]. URL: https://arxiv.org/abs/2306.16636.

[80] Colin White et al. "LiveBench: A Challenging, Contamination-Free LLM Benchmark". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[54] Long Ouyang 等. “Training language models to follow instructions with human feedback”. 载于: *Advances in neural information processing systems* 35 (2022), 第 27730–27744 页. [55] Bowen Peng 等. “Yarn: Efficient conte xt window extension of large language models”. 载于: *arXiv preprint arXiv:2309.00071* (2023). [56] Long Phan 等 . *Humanity's Last Exam*. 2025. arXiv: 2501.14249 [cs.LG]. URL: https://arxiv.org/abs/2501.14249. [57] Penghui Qi 等. “Zero bubble pipeline parallelism”. 载于: *arXiv preprint arXiv:2401.10241* (2023). [58] Yujia Qin 等. “Toolll m: Facilitating large language models to master 16000+ real-world apis”. 载于: *arXiv preprint arXiv:2307.16789* (2 023). [59] Qwen 等. *Qwen2.5 Technical Report*. 2025. arXiv: 2412.15115 [cs.CL]. URL: https://arxiv.org/abs/2412.1 5115. [60] Samyam Rajbhandari 等. “Zero: Memory optimizations toward training trillion parameter models”. 载 于: *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 20 20, 第 1–16 页. [61] David Rein 等. “Gpqa: A graduate-level google-proof q&a benchmark”. 载于:
First Conference on Language Modeling. 2024. [62] Keisuke Sakaguchi 等. “Winogrande: An adversarial winogra d schema challenge at scale”. 载于: *Communications of the ACM* 64.9 (2021), 第 99–106 页. [63] David Silver 与 R ichard S Sutton. “Welcome to the era of experience”. 载于: *Google AI* 1 (2025). [64] Ved Sirdeshmukh 等.
MultiChallenge: A Realistic Multi-Turn Conversation Evaluation Benchmark Challenging to Frontier LLMs. 2025. arXi v: 2501.17399 [cs.CL]. URL: https://arxiv.org/abs/2501.17399. [65] Giulio Starace 等. “PaperBench: Evaluating AI ’ s Ability to Replicate AI Research”. 载于: *arXiv preprint arXiv:2504.01848* (2025). [66] Hao Sun 等.
ZeroSearch: Incentivize the Search Capability of LLMs without Searching. 2025. arXiv: 2505.04588 [cs.CL]. URL: https://arxiv.org/abs/2505.04588. [67] Mirac Suzgun 等.
Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. 2022. arXiv: 2210.09261 [cs.CL]. UR L: https://arxiv.org/abs/2210.09261. [68] Manveer Singh Tamber 等. “Benchmarking LLM Faithfulness in RAG wit h Evolving Leaderboards”. 载于: *arXiv preprint arXiv:2505.04847* (2025). [69] Gemma Team 等. “Gemma 2: Im proving open language models at a practical size”. 载于: *arXiv preprint arXiv:2408.00118* (2024). [70] LlaMA Tea m. *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation — ai.meta.com*. https://ai.meta.co m/blog/llama-4-multimodal-intelligence/. [访问日期 15-07-2025]. [71] The Terminal-Bench Team.
Terminal-Bench: A Benchmark for AI Agents in Terminal Environments. 2025 年 4 月. URL: https://github.com/laude-institute/terminal-bench. [72] Ashish Vaswani 等. “Attention is All you Need”. 载于:
Advances in Neural Information Processing Systems. 由 I. Guyon 等编. 第 30 卷. Curran Associates, Inc., 2017. UR L: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. [73] V ectara. *Hallucination Evaluation Model (Revision 7437011)*. 2024. URL: https://huggingface.co/vectara/hallucinati on_evaluation_model. [74] Joshua Vendrow 等. “Do large language model benchmarks test reliability?” 载于: *arXiv preprint arXiv:2502.03461* (2025). [75] Yizhong Wang 等. “Self-instruct: Aligning language models with se lf-generated instructions”. 载于: *arXiv preprint arXiv:2212.10560* (2022). [76] Yubo Wang 等.
MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. 2024. arXiv: 2406.0157 4 [cs.CL]. URL: https://arxiv.org/abs/2406.01574. [77] Zhexu Wang 等.
OJBench: A Competition Level Code Benchmark For Large Language Models. 2025. arXiv: 2506.16395 [cs.CL]. UR L: https://arxiv.org/abs/2506.16395. [78] Jason Wei 等. “Measuring short-form factuality in large language models ”. 载于: *arXiv preprint arXiv:2411.04368* (2024). [79] Tianwen Wei 等.
CMATH: Can Your Language Model Pass Chinese Elementary School Math Test? 2023. arXiv: 2306.16636 [cs.CL] . URL: https://arxiv.org/abs/2306.16636. [80] Colin White 等. “LiveBench: A Challenging, Contamination-Free LL M Benchmark”. 载于: *The Thirteenth International Conference on Learning Representations*. 2025.

[81] Mitchell Wortsman et al. "Small-scale proxies for large-scale transformer training instabilities, 2023". In: *URL https://arxiv. org/abs/2309.14322* ().

[82] Can Xu et al. *WizardLM: Empowering large pre-trained language models to follow complex instructions*. 2025. arXiv: 2304.12244 [cs.CL]. URL: https://arxiv.org/abs/2304.12244.

[83] Zhangchen Xu et al. *KodCode: A Diverse, Challenging, and Verifiable Synthetic Dataset for Coding*. 2025. arXiv: 2503.02951 [cs.LG]. URL: https://arxiv.org/abs/2503.02951.

[84] John Yang et al. *SWE-smith: Scaling Data for Software Engineering Agents*. 2025. arXiv: 2504.21798 [cs.SE]. URL: https://arxiv.org/abs/2504.21798.

[85] Shunyu Yao et al. "tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains". In: *arXiv preprint arXiv:2406.12045* (2024).

[86] Daoguang Zan et al. "Multi-swe-bench: A multilingual benchmark for issue resolving". In: *arXiv preprint arXiv:2504.02605* (2025).

[87] Eric Zelikman et al. "Star: Bootstrapping reasoning with reasoning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15476–15488.

[88] Rowan Zellers et al. "Hellaswag: Can a machine really finish your sentence?" In: *arXiv preprint arXiv:1905.07830* (2019).

[89] Wanjun Zhong et al. "Agieval: A human-centric benchmark for evaluating foundation models". In: *arXiv preprint arXiv:2304.06364* (2023).

[90] Jeffrey Zhou et al. "Instruction-Following Evaluation for Large Language Models". In: *ArXiv* abs/2311.07911 (2023). URL: https://arxiv.org/abs/2311.07911.

[91] Qin Zhu et al. *AutoLogi: Automated Generation of Logic Puzzles for Evaluating Reasoning Abilities of Large Language Models*. 2025. arXiv: 2502.16906 [cs.CL]. URL: https://arxiv.org/abs/2502.16906.

[81] Mitchell Wortsman 等。“Small-scale proxies for large-scale transformer training instabilities, 2023”。载于：*URL https://arxiv. org/abs/2309.14322* ()。 [82] Can Xu 等。
WizardLM: Empowering large pre-trained language models to follow complex instructions。2025。arXiv: 2304.12244 [cs.CL]。URL: https://arxiv.org/abs/2304.12244。 [83] Zhangchen Xu 等。
KodCode: A Diverse, Challenging, and Verifiable Synthetic Dataset for Coding。2025。arXiv: 2503.02951 [cs.LG]。URL: https://arxiv.org/abs/2503.02951。 [84] John Yang 等。
SWE-smith: Scaling Data for Software Engineering Agents。2025。arXiv: 2504.21798 [cs.SE]。URL: https://arxiv.org/abs/2504.21798。 [85] Shunyu Yao 等。“tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains”。载于：*arXiv preprint arXiv:2406.12045* (2024)。 [86] Daoguang Zan 等。“Multi-swe-bench: A multilingual benchmark for issue resolving”。载于：*arXiv preprint arXiv:2504.02605* (2025)。 [87] Eric Zelikman 等。“Star: Bootstrapping reasoning with reasoning”。载于：*Advances in Neural Information Processing Systems* 35 (2022)，第 15476–15488 页。 [88] Rowan Zellers 等。“Hellaswag: Can a machine really finish your sentence?”载于：*arXiv preprint arXiv:1905.07830* (2019)。 [89] Wanjun Zhong 等。“Agieval: A human-centric benchmark for evaluating foundation models”。载于：*arXiv preprint arXiv:2304.06364* (2023)。 [90] Jeffrey Zhou 等。“Instruction-Following Evaluation for Large Language Models”。载于：*ArXiv* abs/2311.07911 (2023)。URL: https://arxiv.org/abs/2311.07911。 [91] Qin Zhu 等。
AutoLogi: Automated Generation of Logic Puzzles for Evaluating Reasoning Abilities of Large Language Models。2025。arXiv: 2502.16906 [cs.CL]。URL: https://arxiv.org/abs/2502.16906。

Appendix

A Contributions

The listing of authors is in alphabetical order based on their last names. Names marked with an asterisk (*) indicate people who are no longer part of our team.

Yifan Bai
Yiping Bao
Guanduo Chen
Jiahao Chen
Ningxin Chen
Ruijue Chen
Yanru Chen
Yuankun Chen
Yutian Chen
Zhuofu Chen*
Jialei Cui
Hao Ding
Mengnan Dong
Ang'ang Du
Chenzhuang Du
Dikang Du
Yulun Du
Yu Fan
Yichen Feng
Kelin Fu
Bofei Gao
Hongcheng Gao
Tong Gao
Xinran Gu
Longyu Guan
Haiqing Guo*
Jianhang Guo
Xiaoru Hao
Tianhong He
Weiran He
Wenyang He
Chao Hong
Yangyang Hu
Zhenxing Hu
Weixiao Huang
Zhiqi Huang
Zihao Huang
Tao Jiang
Zhejun Jiang
Xinyi Jin
Yongsheng Kang*
Guokun Lai
Cheng Li
Fang Li
Haoyang Li
Ming Li
Wentao Li
Yanhao Li
Yiwei Li
Zhaowei Li
Zheming Li
Hongzhan Lin*
Xiaohan Lin
Zongyu Lin
Chengyin Liu
Chenyu Liu
Hongzhang Liu
Jingyuan Liu*
Junqi Liu
Liang Liu
Shaowei Liu
T.Y. Liu
Tianwei Liu
Weizhou Liu
Yangyang Liu
Yibo Liu
Yiping Liu
Yue Liu
Zhengying Liu
Enzhe Lu
Lijun Lu
Shengling Ma
Xinyu Ma
Yingwei Ma
Shaoguang Mao
Jie Mei
Xin Men
Yibo Miao
Siyuan Pan
Yebo Peng
Ruoyu Qin
Bowen Qu
Zeyu Shang
Lidong Shi
Shengyuan Shi
Feifan Song
Jianlin Su
Zhengyuan Su
Xinjie Sun*
Flood Sung
Heyi Tang
Jiawen Tao
Qifeng Teng
Chensi Wang
Dinglu Wang
Feng Wang
Haiming Wang
Jianzhou Wang*
Jiaxing Wang
Jinhong Wang
Shengjie Wang
Shuyi Wang
Yao Wang
Yejie Wang
Yiqin Wang
Yuxin Wang
Yuzhi Wang
Zhaoji Wang
Zhengtao Wang
Zhexu Wang
Chu Wei
Qianqian Wei
Wenhao Wu
Xingzhe Wu
Yuxin Wu
Chenjun Xiao
Xiaotong Xie
Weimin Xiong*
Boyu Xu
Jing Xu*
Jinjing Xu
L.H. Xu
Lin Xu
Suting Xu
Weixin Xu
Xinran Xu
Yangchuan Xu
Ziyao Xu
Junjie Yan
Yuzi Yan
Xiaofei Yang
Ying Yang
Zhen Yang
Zhilin Yang
Zonghan Yang
Haotian Yao
Xingcheng Yao
Wenjie Ye
Zhuorui Ye
Bohong Yin
Longhui Yu
Enming Yuan
Hongbang Yuan*
Mengjie Yuan
Haobing Zhan
Dehao Zhang
Hao Zhang
Wanlu Zhang
Xiaobin Zhang
Yangkun Zhang
Yizhi Zhang
Yongting Zhang
Yu Zhang
Yutao Zhang
Yutong Zhang
Zheng Zhang
Haotian Zhao
Yikai Zhao
Huabin Zheng
Shaojie Zheng
Jianren Zhou
Xinyu Zhou
Zaida Zhou
Zhen Zhu
Weiyu Zhuang
Xinxing Zu
Kimi K2

附录

A 贡献

作者名单按姓氏字母顺序排列。名字后带星号（*）表示该成员已不在我们团队中。

Yifan Bai
Yiping Bao
Guanduo Chen
Jiahao Chen
Ningxin Chen
Ruijue Chen
Yanru Chen
Yuankun Chen
Yutian Chen
Zhuofu Chen*
Jialei Cui
Hao Ding
Mengnan Dong
Ang'ang Du
Chenzhuang Du
Dikang Du
Yulun Du
Yu Fan
Yichen Feng
Kelin Fu
Bofei Gao
Hongcheng Gao
Tong Gao
Xinran Gu
Longyu Guan
Haiqing Guo*
Jianhang Guo
Xiaoru Hao
Tianhong He
Weiran He
Wenyang He
Chao Hong
Yangyang Hu
Zhenxing Hu
Weixiao Huang
Zhiqi Huang
Zihao Huang
Tao Jiang
Zhejun Jiang
Xinyi Jin
Yongsheng Kang*
Guokun Lai
Cheng Li
Fang Li
Haoyang Li
Ming Li
Wentao Li
Yanhao Li
Yiwei Li
Zhaowei Li
Zheming Li
Hongzhan Lin*
Xiaohan Lin
Zongyu Lin
Chengyin Liu
Chenyu Liu
Hongzhang Liu
Jingyuan Liu*
Junqi Liu
Liang Liu
Shaowei Liu
T.Y. Liu
Tianwei Liu
Weizhou Liu
Yangyang Liu
Yibo Liu
Yiping Liu
Yue Liu
Zhengying Liu
Enzhe Lu
Lijun Lu
Shengling Ma
Xinyu Ma
Yingwei Ma
Shaoguang Mao
Jie Mei
Xin Men
Yibo Miao
Siyuan Pan
Yebo Peng
Ruoyu Qin
Bowen Qu
Zeyu Shang
Lidong Shi
Shengyuan Shi
Feifan Song
Jianlin Su
Zhengyuan Su
Xinjie Sun*
Flood Sung
Heyi Tang
Jiawen Tao
Qifeng Teng
Chensi Wang
Dinglu Wang
Feng Wang
Haiming Wang
Jianzhou Wang*
Jiaxing Wang
Jinhong Wang
Shengjie Wang
Shuyi Wang
Yao Wang
Yejie Wang
Yiqin Wang
Yuxin Wang
Yuzhi Wang
Zhaoji Wang
Zhengtao Wang
Zhexu Wang
Chu Wei
Qianqian Wei
Wenhao Wu
Xingzhe Wu
Yuxin Wu
Chenjun Xiao
Xiaotong Xie
Weimin Xiong*
Boyu Xu
Jing Xu*
Jinjing Xu
L.H. Xu
Lin Xu
Suting Xu
Weixin Xu
Xinran Xu
Yangchuan Xu
Ziyao Xu
Junjie Yan
Yuzi Yan
Xiaofei Yang
Ying Yang
Zhen Yang
Zhilin Yang
Zonghan Yang
Haotian Yao
Xingcheng Yao
Wenjie Ye
Zhuorui Ye
Bohong Yin
Longhui Yu
Enming Yuan
Hongbang Yuan*
Mengjie Yuan
Haobing Zhan
Dehao Zhang
Hao Zhang
Wanlu Zhang
Xiaobin Zhang
Yangkun Zhang
Yizhi Zhang
Yongting Zhang
Yu Zhang
Yutao Zhang
Yutong Zhang
Zheng Zhang
Haotian Zhao
Yikai Zhao
Huabin Zheng
Shaojie Zheng
Jianren Zhou
Xinyu Zhou
Zaida Zhou
Zhen Zhu
Weiyu Zhuang
Xinxing Zu
Kimi K2

B Token Template of Tool Calling

There are three components in the token structure for tool-calling:

- **Tool declaration message**: defines the list of available tools and the schema of the arguments;
- **Tool invoking section in assistant message**: encodes the model's request to invoke tools;
- **Tool result message**: encapsulates the invoked tool's execution result.

The raw tokens of the tool declaration message are formatted as follows:

```
<|im_begin|>
tool_declare
<|im_middle|>
# Tools

{{ tool declaration content }}
<|im_end|>
```

The blue highlighted marks represent special tokens, and the green part, quoted by brackets, is the tool declaration content. We use TypeScript to express the tool declaration content, since TypeScript is a concise language with a comprehensive type system, able to express the types and constraints of tool parameters with brief text. The code 1 shows an example for two simple tools in JSON format compatible with OpenAI's chat completion API, as a comparison, the same tools defined in TypeScript (listed in Code 2) is much shorter. To improve compatibility, part of our training data also uses JSON as the tool declaration language, so that 3rd-party frameworks need not additional development to support our tool calling scheme.

Listing 1: Tool definition with JSON in OpenAI compatible API

```
[{
  "type": "function",
  "function": {
    "name": "get_weather",
    "description": "Get weather for a location and date",
    "parameters": {
      "type": "object",
      "properties": {
        "location": {
          "type": "string",
          "description": "City and country e.g. Beijing, China"
        },
        "date": {
          "type": "string",
          "description": "Date to query, format in '%Y-%m-%d'"
        }
      },
      "required": [
        "location"
      ]
    }
  }
},
{
  "type": "function",
  "function": {
    "name": "Calculator",
    "description": "Simple calculator",
    "parameters": {
      "properties": {
        "expr": {
          "type": "string",
          "description": "Arithmetic expression in javascript"
        }
      },
```

B 工具调用的 Token 模板

工具调用的 token 结构中有三个组成部分：

- 工具声明消息：定义可用工具的列表以及参数的模式；
- 助手消息中的工具调用部分：编码了模型调用工具的请求；
- 工具结果消息：封装了被调用工具的执行结果。

工具声明消息的原始标记格式如下：

```
<|im_begin|>
tool_declare
<|im_middle|>
# Tools

{{ tool declaration content }}
<|im_end|>
```

蓝色高亮标记表示特殊 token，绿色部分（用括号括起）是工具声明内容。我们使用 TypeScript 来表达工具声明内容，因为 TypeScript 是一种简洁的语言，拥有完备的类型系统，能够用简短文本表达工具参数的类型与约束。代码 1 展示了两个简单工具的示例，以兼容 OpenAI 聊天补全 API 的 JSON 格式呈现；相比之下，用 TypeScript 定义的相同工具（见代码 2）要简短得多。为提高兼容性，我们的部分训练数据也使用 JSON 作为工具声明语言，这样第三方框架无需额外开发即可支持我们的工具调用方案。

清单 1：使用 JSON 的 OpenAI 兼容工具定义 API

```
[{ "type": "function", "function": { "name": "get_weather", "description": "获取某地点某天的天
气", "parameters": { "type": "object", "properties": { "location": { "type": "string", "description": "
城市和国家，例如 Beijing, China" }, "date": { "type": "string", "description": "要查询的日期,
格式为 ‘%Y-%m-%d’ " } }, "required": ["location"] } } }, { "type": "function", "function": { "n
ame": "Calculator", "description": "简易计算器", "parameters": { "properties": { "expr": { "type":
"string", "description": "JavaScript 算术表达式" } }
```

```
      "type": "object"
    }
  }
}]
```

Listing 2: Tool definition in TypeScript

```
namespace functions {
// Get weather for a location and date
type get_weather = (_: {
  // City and country e.g. Beijing, China
  location: string,
  // Date to query, format in '%Y-%m-%d'
  date?: string
}) => any;
// Simple calculator
type Calculator = (_: {
  // Arithmetic expression in javascript
  expr?: string
}) => any;
}
```

The token template of the tool invoking section in the model's response messages is listed as follows:

```
<tool_call_section_begin|>
<|tool_call_begin|>
// call_id part
functions.{{tool name}}:{{counter}}
<|tool_arguments_begin|>
{{ json serialized call arguments }}
<|tool_call_end|>
<|tool_call_begin|>
// more tool calls
<|tool_call_end|>
<|tool_call_section_end|>
```

As shown in the template, we support parallel tool calling by placing multiple tool calls in a single response turn. Each tool call has a unique call id, formatted as `functions.{tool-name}:{counter}`, where `tool-name` is the name of the tool, and `counter` is an auto-increasing counter of all tool calls starting from 0 in the dialog.

During inference, the model may occasionally generate unexpected tokens, leading to format errors when parsing a tool call. To solve this issue, we developed a constrained decoding module named *enforcer*, inspired by lm-format-enforcer[6]. When a `<tool_call_section_begin|>` token is generated, it ensures that the upcoming tool-related tokens follow the predefined template, and the JSON argument string follows the declared schema.

The tool result message is simply a text message encoded with the tool's call id and the corresponding results.

```
<|im_begin|>
tool
<|im_middle|>
## Results of {{call_id}}
{{ execution result content }}
<|im_end|>
```

C Evaluation Details

Coding Tasks. We evaluate Kimi-K2-Instruct's capabilities on competitive coding benchmarks, LiveCodeBench and OJBench, where Kimi-K2-Instruct attains superior performance with scores of 53.7% and 27.1%, respectively. This excellence spans both medium-level coding challenges, such as LeetCode and AtCoder, and hard-level contests like NOI and ICPC, outperforming leading open-source and proprietary models. For multilingual programming proficiency, we employ MultiPL-E, covering languages including C++, C#, Java, JavaScript, PHP, Go, Kimi-K2-Instruct surpasses top

[6] https://github.com/noamgat/lm-format-enforcer

"类型": "对象" } } }]

清单 2：TypeScript 中的工具定义

命名空间 functions { // 根据地点和日期获取天气 type get_w
eather = （_: { // 城市和国家，例如 Beijing, China location : s
tring , // 要查询的日期，格式为 ‘%Y-%m-%d’ date ?: strin
g }） => any ; // 简易计算器 type Calculator = （_: { // JavaSc
ript 算术表达式 expr ?: string }） => any ; }

模型响应消息中工具调用部分的 token 模板如下所示：

<|

<tool_call_section_begin|>tool_call_begin|> // call_id 部分 functions.{{tool name}}:{{cou nter}} <|tool_arguments_begin|> {{ json 序 列化的调用参数 }} <|tool_call_end|>tool_ca ll_begin|> // 更多工具调用 <|tool_call_end |>tool_call_section_end|>

<|

如模板所示，我们通过在单个响应回合中放置多个工具调用来支持并行工具调用。每个工具调用都有一个唯一的调用 id，格式为 functions.{tool-name}:{counter}，其中 tool-name 是工具的名称，counter 是从对话中所有工具调用从 0 开始自动递增的计数器。

在推理过程中，模型偶尔会生成意外的标记，导致解析工具调用时出现格式错误。为解决这一问题，我们开发了一个受 lm-format-enforcer[6] 启发的约束解码模块，名为 *enforcer*。当生成 <tool_call_section_begin|> 标记时，它会确保接下来的工具相关标记遵循预定义模板，且 JSON 参数字符串符合声明的架构。

工具结果消息只是一个文本消息，使用工具的调用 ID 和相应结果进行编码。

```
<|im_begin|>
tool
<|im_middle|>
## Results of {{call_id}}
{{ execution result content }}
<|im_end|>
```

C 评估详情

编码任务。我们在竞技编程基准 LiveCodeBench 和 OJBench 上评估 Kimi-K2-Instruct 的能力，Kimi-K2-Instruct 分别取得 53.7% 和 27.1% 的优异成绩。这一卓越表现涵盖中等难度的编码挑战（如 LeetCode 和 AtCoder）以及高难度竞赛（如 NOI 和 ICPC），超越了领先的开源和专有模型。在多语言编程能力方面，我们使用 MultiPL-E，涵盖 C++、C#、Java、JavaScript、PHP、Go 等语言，Kimi-K2-Instruct 超越了顶级

[6] https://github.com/noamgat/lm-format-enforcer

open-source models with an accuracy of 85.7%, compared with 83.1% for DeepSeek-V3-0324 and 78.2% for Qwen3-235B-A22B. In software engineering tasks, Kimi-K2-Instruct demonstrates robust performance on SWE-bench Verified (Python), SWE-lancer (Python), SWE-bench Multilingual, and Multi-SWE-bench datasets. It significantly outperforms open-source counterparts in resolving real-world code repository issues and notably narrows the performance gap with proprietary models. For example:

- SWE-bench Verified (multiple attempts): 71.6% (Kimi-K2-Instruct) vs. 80.2% (Claude 4 Sonnet)
- SWE-bench Multilingual: 47.3% (Kimi-K2-Instruct) vs. 51.0% (Claude 4 Sonnet)
- SWE-lancer: 39.1% (Kimi-K2-Instruct) vs. 40.8% (Claude 4 Sonnet)

On PaperBench, Kimi-K2-Instruct achieves an accuracy of 27.8%, closely matching GPT-4.1 and outperforming DeepSeek-V3-0324 (12.2%) and Qwen3-235B-A22B (8.2%) by a substantial margin. In terminal interaction tasks measured by TerminalBench, Kimi-K2-Instruct attains 25.0% using the default Terminus framework and rises to 30% within Moonshot's in-house agentic framework, underscoring its capabilities in real-world agentic programming scenarios. Moreover, on the Aider-Polyglot benchmark, Kimi-K2-Instruct attains a 60.0% accuracy while employing rigorous decontamination procedures, further illustrating its strength and reliability across diverse coding environments.

Tool Use Tasks. We evaluate multi-turn tool use with two complementary suites: τ^2-Bench and ACEBench. τ^2-Bench extends the original τ-bench single-control setup to a *dual-control* environment in which both the agent and an LLM-simulated user have constrained tool affordances over a shared state, adding a realistic Telecom troubleshooting domain alongside the prior Airline/Retail TAU tasks and enabling analysis of coordination vs. pure reasoning. ACEBench is a large bilingual (En/Zh) API-grounded benchmark (4.5K APIs across 8 domains; 2K annotated eval items) partitioned into NORMAL (basic/personalized/atomic), SPECIAL (imperfect or out-of-scope inputs), and AGENT (scenario-driven multi-turn, multi-step sandbox) tracks with automated grading of calls and outcomes. All models run in non-thinking mode; we set the temperature to 0.0, use deterministic tool adapters, score τ^2 Airline/Retail/Telecom under Avg@4 seeds with Pass@1/4, and report overall on ACEBench English. Kimi-K2-Instruct averages 66.1 micro Pass@1 across τ^2 vs DeepSeek-V3-0324 48.8 / Qwen3-235B-A22B 37.3. On ACEBench Overall Kimi-K2-Instruct scores 76.5 vs DeepSeek 72.7 / Qwen 70.5 and remains competitive with GPT-4.1 (80.1).

Math & STEM & Logical Tasks. For Math tasks, Kimi-K2-Instruct achieves consistently strong performance, averaging over Geimini-2.5-Flash by 5.3 percentage points, over DeepSeek-V3-0324 by 5.5 points and over GPT4.1 by 15.8 points. For example, on AIME 2024, Kimi-K2-Instruct scores 69.6%, outperforming another two top open-source models by a large margin, DeepSeek-V3-0324 by 10.2 points and Qwen3-235B-A22B by 29.5 points. In STEM evaluations, Kimi-K2-Instruct achieves 75.1% on GPQA-Diamond, outperforming DeepSeek-V3-0324 (68.4%) and all non-thinking baselines by at least 5 percentage points. On SuperGPQA, it also exceeds the previous best open-source model, DeepSeek-V3-0324, by 3.5 points. Kimi-K2-Instruct also surpasses the other two leading models in logical reasoning. It achieves 89.0% on ZebraLogic and 89.5% on AutoLogi, exceeding DeepSeek-V3-0324 (84.0%, 88.9%) and substantially outperforming Qwen3-235B-A22B (37.7%, 83.3%).

General Tasks. Kimi-K2-Instruct ties DeepSeek-V3-0324 on MMLU and MMLU-Pro, and takes the lead on MMLU-Redux with a 92.7 EM score—slightly ahead of GPT-4.1 (92.4) and just 1.5 points behind Claude-Opus-4. Beyond multiple-choice tasks, the model achieves 31.0% accuracy on the short-answer SimpleQA—3.3 points above DeepSeek-V3-0324 and more than twice that of Qwen3-235B-A22B—though still below GPT-4.1 (42.3%). On the adversarial free-response LiveBench (2024-11-25 snapshot), it reaches 76.4%, surpassing Claude-Sonnet 4 (74.8%) and leading Gemini 2.5 Flash Preview by 8.6 points. Across this challenging triad measuring breadth, depth, and robustness of world knowledge, Kimi-K2-Instruct secures a top-tier position among open-source models. We evaluate instruction-following with IFEval and Multi-Challenge. On IFEval, Kimi-K2-Instruct scores 89.8%, higher than DeepSeek-V3-0324 (81.1%) and GPT-4.1 (88.0%). On Multi-Challenge, which involves multi-turn dialogues with conflicting instructions, it achieves 54.1%, outperforming DeepSeek-V3-0324 (31.4%), GPT-4.1 (36.4%), and Claude-Opus-4 (49.0%). These results demonstrate that Kimi-K2-Instruct integrates strong factual knowledge with consistent instruction adherence across both single- and multi-turn settings, supporting robust and reliable real-world deployment.

Long Context and Factuality Tasks. To evaluate the factuality of Kimi-K2-Instruct, we employ three benchmarks: FACTS Grounding, which measures adherence to provided documents using the proprietary models GPT-4o, Gemini 1.5 Pro and Claude 3.5 Sonnet; HHEM, which assesses summarization quality via the open-source HHEM-2.1-Open judge; and FaithJudge, which analyzes faithfulness in RAG tasks with o3-mini as the judge. Kimi-K2-Instruct scores 88.5 on FACTS Grounding, substantially outperforming all open-source rivals and even surpassing the closed-source Gemini 2.5 Flash. With HHEM-2.1-Open it achieves a hallucination rate of 1.1 %, reported in the tables as 1 minus the

开源模型的准确率为 85.7%，相比之下，DeepSeek-V3-0324 为 83.1%，Qwen3-235B-A22B 为 78.2%。在软件工程任务中，Kimi-K2-Instruct 在 SWE-bench Verified（Python）、SWE-lancer（Python）、SWE-bench Multilingual 和 Multi-SWE-bench 数据集上均展现出稳健的性能。它在解决真实代码仓库问题方面显著优于开源对手，并明显缩小了与专有模型的性能差距。例如：

- SWE-bench Verified（多次尝试）：71.6%（Kimi-K2-Instruct）vs. 80.2%（Claude 4 Sonn等)
- SWE-bench 多语言：47.3%（Kimi-K2-Instruct）vs. 51.0%（Claude 4 Sonnet）
- SWE-lancer：39.1%（Kimi-K2-Instruct）vs. 40.8%（Claude 4 Sonnet）

在 PaperBench 上，Kimi-K2-Instruct 的准确率达到 27.8%，与 GPT-4.1 非常接近，并大幅超越 DeepSeek-V3-0324（12.2%）和 Qwen3-235B-A22B（8.2%）。在通过 TerminalBench 衡量的终端交互任务中，Kimi-K2-Instruct 在默认 Terminus 框架下取得 25.0%，在 Moonshot 自研的 agentic 框架中进一步提升至 30%，凸显其在真实 agentic 编程场景中的能力。此外，在 Aider-Polyglot 基准测试中，Kimi-K2-Instruct 在采用严格去污染流程的情况下达到 60.0% 的准确率，进一步展现了其在多样化编程环境中的实力与可靠性。

工具使用任务。我们通过两个互补的套件评估多轮工具使用：τ^2-Bench 和 ACEBench。τ^2-Bench 将原始的 τ-bench 单控制设置扩展为 *dual-control* 环境，其中智能体与 LLM 模拟用户都对共享状态拥有受限的工具能力，新增真实的电信故障排查领域，并保留之前的航空/零售 TAU 任务，从而支持对协作与纯推理的分析。ACEBench 是一个大型双语（英/中）基于 API 的基准（8 个领域共 4.5K API；2K 条带标注的评估项），划分为 NORMAL（基础/个性化/原子）、SPECIAL（不完美或超出范围的输入）和 AGENT（场景驱动的多轮、多步沙盒）赛道，并自动对调用与结果进行评分。所有模型均以非思考模式运行；我们将温度设为 0.0，使用确定性工具适配器，在 Avg@4 种子下对 τ^2 航空/零售/电信评分，采用 Pass@1/4，并在 ACEBench 英文部分报告总体结果。Kimi-K2-Instruct 在 τ^2 上平均 micro Pass@1 为 66.1，而 DeepSeek-V3-0324 为 48.8 / Qwen3-235B-A22B 为 37.3。在 ACEBench 总体得分上，Kimi-K2-Instruct 为 76.5，DeepSeek 为 72.7 / Qwen 为 70.5，仍与 GPT-4.1（80.1）保持竞争力。

数学与STEM及逻辑任务。在数学任务中，Kimi-K2-Instruct始终保持强劲表现，平均领先Geimini-2.5-Flash 5.3个百分点，领先DeepSeek-V3-0324 5.5个百分点，领先GPT4.1 15.8个百分点。例如，在AIME 2024上，Kimi-K2-Instruct得分69.6%，大幅领先另外两个顶级开源模型：领先DeepSeek-V3-0324 10.2分，领先Qwen3-235B-A22B 29.5分。在STEM评估中，Kimi-K2-Instruct在GPQA-Diamond上取得75.1%，领先DeepSeek-V3-0324（68.4%）和所有非思考基线至少5个百分点。在SuperGPQA上，它也超过此前最佳开源模型DeepSeek-V3-0324 3.5分。Kimi-K2-Instruct在逻辑推理方面也超越另外两个领先模型：在ZebraLogic上取得89.0%，在AutoLogi上取得89.5%，均超过DeepSeek-V3-0324（84.0%、88.9%），并大幅领先Qwen3-235B-A22B（37.7%、83.3%）。

通用任务。Kimi-K2-Instruct 在 MMLU 和 MMLU-Pro 上与 DeepSeek-V3-0324 持平，并在 MMLU-Redux 上以 92.7 的 EM 分领先，略高于 GPT-4.1（92.4），仅落后 Claude-Opus-4 1.5 分。在选择题之外，该模型在简答题 SimpleQA 上达到 31.0% 的准确率——比 DeepSeek-V3-0324 高 3.3 分，是 Qwen3-235B-A22B 的两倍多——但仍低于 GPT-4.1（42.3%）。在对抗性自由回答 LiveBench（2024-11-25 快照）上，它取得 76.4%，超越 Claude-Sonnet 4（74.8%），领先 Gemini 2.5 Flash Preview 8.6 分。在这组衡量世界知识广度、深度与鲁棒性的严苛三元测试中，Kimi-K2-Instruct 稳居开源模型第一梯队。我们使用 IFEval 和 Multi-Challenge 评估指令遵循能力。在 IFEval 上，Kimi-K2-Instruct 得分 89.8%，高于 DeepSeek-V3-0324（81.1%）和 GPT-4.1（88.0%）。在涉及多轮对话且指令冲突的 Multi-Challenge 上，它取得 54.1%，优于 DeepSeek-V3-0324（31.4%）、GPT-4.1（36.4%）和 Claude-Opus-4（49.0%）。这些结果表明，Kimi-K2-Instruct 在单轮与多轮场景下均能将扎实的事实知识与一致的指令遵循相结合，支持稳健可靠的真实世界部署。

长上下文与事实性任务。为评估 Kimi-K2-Instruct 的事实性，我们采用三项基准：FACTS Grounding，通过专有模型 GPT-4o、Gemini 1.5 Pro 和 Claude 3.5 Sonnet 衡量对给定文档的遵循程度；HHEM，通过开源裁判 HHEM-2.1-Open 评估摘要质量；FaithJudge，使用 o3-mini 作为裁判分析 RAG 任务中的忠实度。Kimi-K2-Instruct 在 FACTS Grounding 上得分 88.5，大幅领先所有开源对手，甚至超过闭源的 Gemini 2.5 Flash。在 HHEM-2.1-Open 上，其幻觉率为 1.1%，在表格中以 1 减去该值报告。

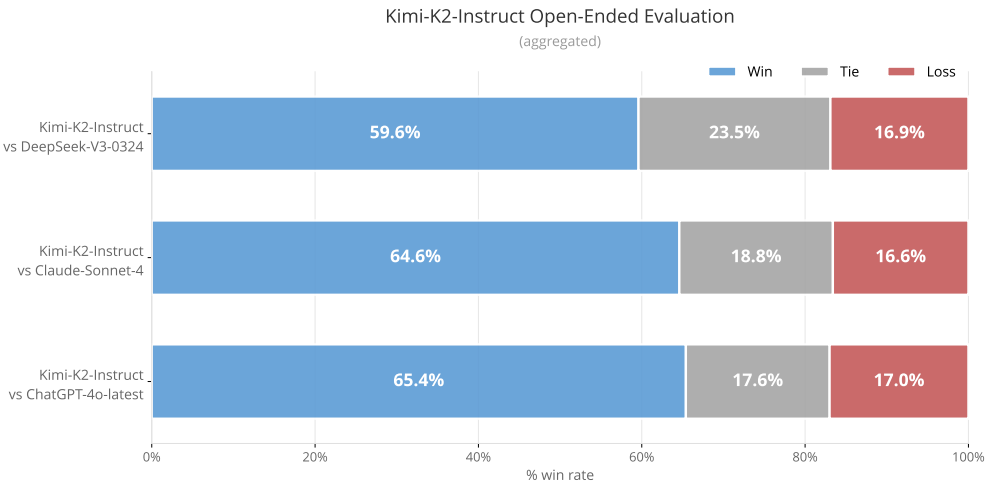


Figure 11: Chinese in-house benchmark evaluation.

rate, i.e. 98.9. On FaithJudge's RAG tasks the hallucination rate is 7.4 %, likewise present as 92.6 for table consistency. For long-context capabilities, Kimi-K2-Instruct outperforms all open source and proprietary models on DROP (93.5%), and exceeds DeepSeek-V3-0324 on retrieval task MRCR (55.0% vs 50.8%). For long-context reasoning tasks FRAMES and LongBench v2, Kimi-K2-Instruct (77.1%, 49.1%) lags slightly behind DeepSeek-V3-0324 by around 2%.

Open-Ended Evaluation Beyond static, closed-ended benchmarks, we evaluate the model's performance on open-ended, nuanced tasks that more closely resemble real-world usage.

For English scenarios, we leverage the Arena-Hard-Auto v2.0 benchmark, which use LLM-as-a-judge protocols to assess generation quality across diverse, open-ended prompts [42]. These evaluations cover a wide range of high-difficulty prompts and are widely recognized in the research community. On Arena-Hard-Auto v2.0, Kimi-K2-Instruct achieves state-of-the-art win-rate on both hard prompts (54.5%) and creative writing tasks (85.0%), outperforming all open-source models and rivaling top proprietary systems such as GPT-4.1 and Claude Sonnet. These results underscore the model's strength in handling complex reasoning and nuanced generation under diverse, unconstrained settings.

However, Arena-Hard-Auto provides limited coverage of Chinese-specific tasks. To address this gap, we developed an in-house held-out benchmark grounded in authentic user queries. To safeguard the integrity of the evaluation, the benchmark data is access-restricted, thereby eliminating the risk of overfitting.

As shown in Figure 11, Kimi-K2-Instruct shows strong performance across all comparisons on Chinese in-house benchmarks. It outperforms ChatGPT-4o-latest with a 65.4% win rate, Claude Sonnet 4 with 64.6%, and DeepSeek-V3-0324 with 59.6%. In all cases, the loss rate stays low (around 17%), indicating that Kimi-K2-Instruct rarely falls behind. The high win rates and consistent margins demonstrate its strong ability on open-ended Chinese tasks.

In addition to controlled evaluations, we also consider real-world user preference through public human assessments. As of July 17, 2025, Kimi-K2-Instruct ranked as the top open-source model and fifth overall on the LMSYS Arena leaderboard[7], based on over 3,000 blind votes from real users. Unlike LLM-as-a-judge protocols, this leaderboard reflects direct human preference on diverse, user-submitted prompts, providing a complementary perspective on practical model performance.

The results on Arena-Hard-Auto, our in-house benchmark and votes from LMSYS Arena collectively offer a comprehensive view of Kimi-K2-Instruct's open-ended capabilities, showing that it is a highly preferred model in real-world user experience across English and Chinese.

D QK-Clip Does Not Impair Model Quality

The QK-Clip design follows a **minimal intervention principle**: it activates only when necessary, and deactivates after training stabilizes. Empirical evidence and analysis converge on its negligible impact on model quality.

[7] https://lmarena.ai/leaderboard/text

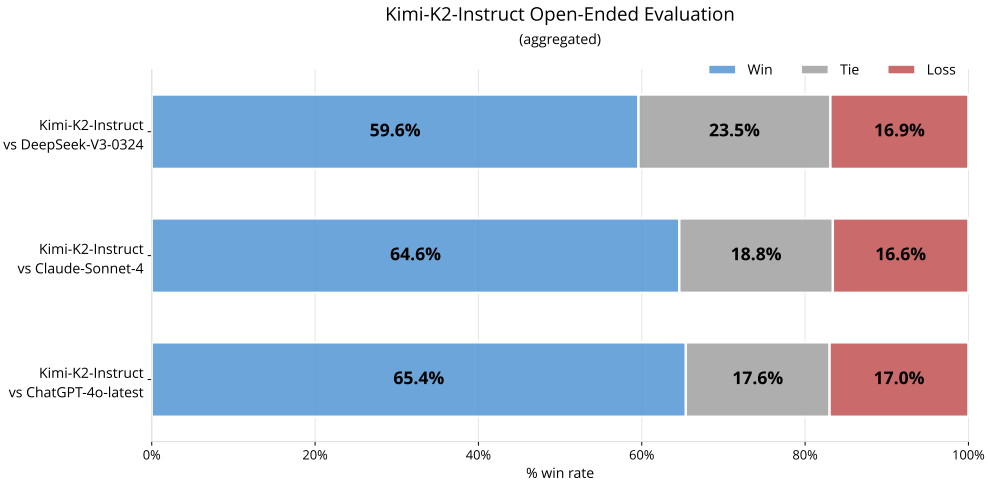


图11：中文内部基准评估。

准确率，即 98.9。在 FaithJudge 的 RAG 任务中，幻觉率为 7.4%，同样以 92.6 的表格一致性呈现。在长文本能力方面，Kimi-K2-Instruct 在 DROP 上超越所有开源和专有模型（93.5%），并在检索任务 MRCR 上超过 DeepSeek-V3-0324（55.0% vs 50.8%）。在长文本推理任务 FRAMES 和 LongBench v2 上，Kimi-K2-Instruct（77.1%、49.1%）略低于 DeepSeek-V3-0324，差距约 2%。

开放式评估 除了静态、封闭式的基准测试外，我们还评估模型在开放式、细致入微的任务上的表现，这些任务更贴近真实世界的使用场景。

对于英文场景，我们采用 Arena-Hard-Auto v2.0 基准，该基准使用 LLM-as-a-judge 协议来评估在多样化、开放式提示下的生成质量 [42]。这些评估涵盖了大量高难度提示，并在研究界广受认可。在 Arena-Hard-Auto v2.0 上，Kimi-K2-Instruct 在困难提示（54.5%）和创意写作任务（85.0%）上均取得了当前最高的胜率，超越了所有开源模型，并与 GPT-4.1 和 Claude Sonnet 等顶级闭源系统相媲美。这些结果凸显了该模型在多样化、无约束场景下处理复杂推理和细腻生成的强大能力。

然而，Arena-Hard-Auto 对中文特定任务的覆盖有限。为弥补这一缺口，我们基于真实用户查询构建了一个内部保留基准。为保障评估的完整性，该基准数据受到访问限制，从而消除了过拟合的风险。

如图 11 所示，Kimi-K2-Instruct 在所有中文内部基准对比中均表现强劲。它以 65.4% 的胜率超越 ChatGPT-4o-latest，以 64.6% 的胜率超越 Claude Sonnet 4，并以 59.6% 的胜率超越 DeepSeek-V3-0324。在所有情况下，其败率均保持在较低水平（约 17%），表明 Kimi-K2-Instruct 很少落后。高胜率与稳定的差距充分展现了其在开放式中文任务上的强大能力。

除了受控评估外，我们还通过公开的人工评测来考察真实世界中的用户偏好。截至 2025 年 7 月 17 日，Kimi-K2-Instruct 在 LMSYS Arena 排行榜[7]上位列开源模型第一、总体第五，这一排名基于超过 3,000 名真实用户的盲投。与 LLM-as-a-judge 协议不同，该排行榜直接反映了人类对多样化用户提交提示的偏好，为模型的实际性能提供了互补视角。

Arena-Hard-Auto 上的结果、我们的内部基准以及来自 LMSYS Arena 的投票共同提供了对 Kimi-K2-Instruct 开放式能力的全面视角，表明它在英语和中文的真实用户体验中是一款极受青睐的模型。

D QK-Clip 不会损害模型质量

QK-Clip 设计遵循最小干预原则：仅在必要时激活，并在训练稳定后停用。实证证据和分析均表明其对模型质量的影响可以忽略不计。

[7] https://lmarena.ai/leaderboard/text

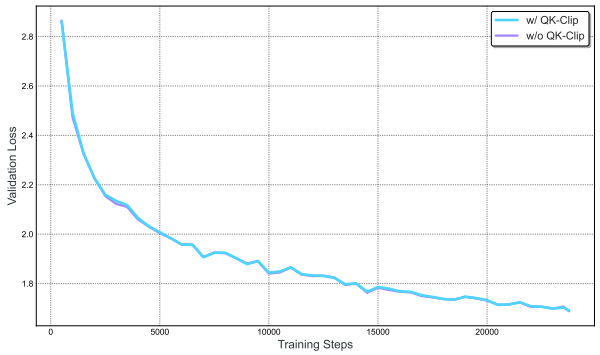


Figure 12: Applying QK-Clip to Muon in a small-scale setting with an aggresive threshold ($\tau = 30$) has negligible impact on loss, indicating that it is a safe and effective method for constraining attention logits.

Small-Scale Ablations We train two small-scale 0.5B activated and 3B total parameters MoE models, one with vanilla Muon and the other with MuonClip using a low clipping threshold ($\tau = 30$). As shown in Figure 12, applying MuonClip has negligible effects on the loss curve, indicating that even aggressive clipping does not impair convergence or training dynamics with MuonClip. This demonstrates that MuonClip is a safe and effective method for bounding attention logits without degrading model performance. Furthermore, evaluation on downstream tasks reveals no statistically significant degradation in performance. These results collectively demonstrate that MuonClip is a safe and effective method for bounding attention logits without compromising model quality.

Self-deactivation In Kimi K2, QK-Clip was only transiently active:

- **Initial 70000 steps:** 12.7% of attention heads triggered QK-Clip for at least once, clamping $S_{\max}$ to 100.
- **Post-70000 steps:** All heads at some point reduced their $S_{\max}$ below 100, rendering QK-Clip inactive.

When QK-Clip is active, it is applied per-head (rather than per-layer) to minimize potential over-regularization on other heads. After training stabilizes, QK-clip is deactivated and has no effect at all.

E Why Muon is More Prone to Logit Explosion

Logit explosion occurs when the largest pre-softmax attention score

$$S_{\max} = \max_{i,j}(q_i \cdot k_j) \tag{1}$$

grows unboundedly during training. Since

$$|q_i \cdot k_j| \leq \|q_i\|\|k_j\| \leq \|x_i\|\|x_j\|\|\mathbf{W}_q\|\|\mathbf{W}_k\|, \tag{2}$$

and RMS-Norm keeps $\|x_i\|\|x_j\|$ bounded, the phenomenon is primarily driven by the growing spectral-norm of $\mathbf{W}_q$ or $\mathbf{W}_k$. Empirically, we found that Muon is more susceptible to logit explosion. We give our hypothesis below.

Structural difference in updates Muon produces a weight update coming from the msign operation; as a result, *all* singular values of the update matrix are equal — its effective rank is full. In contrast, a typical update matrix produced by Adam exhibits a skewed spectrum: a few large singular values dominate, and the effective rank is low. This low-rank assumption for Adam is not new; higher-order muP makes the same assumption.

Such phenomenon is verified on the 16 B Moonlight model, which shows weights trained with Muon exhibit higher *singular-value entropy* (i.e. higher effective rank) than those trained with Adam, corroborating the theoretical intuition.

SVD formulation Let the parameter matrix at step $t-1$ have the singular value decomposition

$$\mathbf{W}_{t-1} = \sum_i \sigma_i \, u_i v_i^\top \tag{3}$$

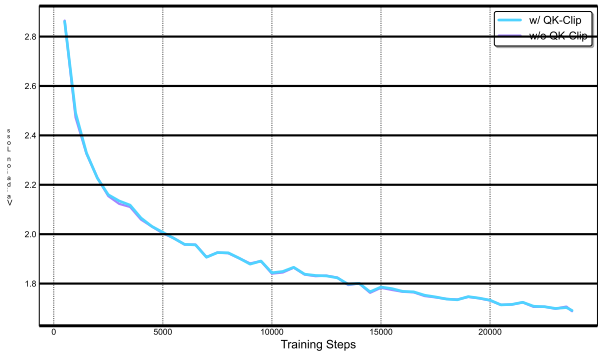


图12：在小规模设置中对Muon应用QK-Clip并使用激进阈值（$\tau = 30$）对损失的影响可以忽略不计，表明这是一种安全且有效的约束注意力logits的方法。

小规模消融实验　我们训练了两个小规模MoE 模型：激活参数 0.5B、总参数 3B，一个使用 vanilla Muon，另一个使用 MuonClip 并采用低裁剪阈值（$\tau = 30$）。如图 12 所示，应用 MuonClip 对损失曲线几乎没有影响，表明即使采用激进的裁剪，也不会损害 MuonClip 的收敛性或训练动态。这证明 MuonClip 是一种安全且有效的方法，可在不降低模型性能的前提下限制注意力 logits。此外，在下游任务上的评估也未发现性能出现统计学意义上的显著下降。这些结果共同表明，MuonClip 是一种安全且有效的方法，可在不牺牲模型质量的情况下限制注意力 logits。

在 Kimi K2 中，QK-Clip 只是短暂激活：

- 初始70000步：12.7%的注意力头至少触发了一次QK-Clip，将$S_{\max}$限制为100。
- 70000步之后：所有头在某个时刻都将它们的$S_{\max}$降至100以下，导致QK-Clip失效。

当 QK-Clip 启用时，它是按头（而非按层）应用的，以尽量减少对其他头的潜在过度正则化。训练稳定后，QK-clip 会被停用，完全不再产生任何影响。

E 为什么 Muon 更容易出现 Logit 爆炸

Logit爆炸发生在最大的pre-softmax注意力分数时

$$S_{\max} = \max_{i,j}(q_i \cdot k_j) \tag{1}$$

在训练过程中无界增长。由于

$$|q_i \cdot k_j| \le \|q_i\|\|k_j\| \le \|x_i\|\|x_j\|\|\mathbf{W}_q\|\|\mathbf{W}_k\|, \tag{2}$$

RMS-Norm 保持 $\|x_i\|\|x_j\|$ 有界，这一现象主要由 $\mathbf{W}_q$ 或 $\mathbf{W}_k$ 的谱范数增长驱动。实验上，我们发现 Muon 更容易出现 logit 爆炸。我们在下面给出假设。

Muon 在更新上的结构差异在于，其权重更新来自 msign 运算；因此，更新矩阵的 *all* 奇异值全部相等——其有效秩为满秩。相比之下，Adam 产生的典型更新矩阵呈现偏斜谱：少数大奇异值占主导，有效秩较低。Adam 的这种低秩假设并非新观点；更高阶的 muP 也采用了相同的假设。

这种现象在 16 B Moonlight 模型上得到了验证，显示使用 Muon 训练的权重具有更高的 *singular-value entropy* (，即更高的有效秩)，与理论直觉相符。

SVD 公式 设步骤 $t-1$ 的参数矩阵具有奇异值分解

$$\mathbf{W}_{t-1} = \sum_i \sigma_i \, u_i v_i^\top \tag{3}$$

We write the update matrices as

$$\Delta \mathbf{W}_t = \sum_j \bar{\sigma}\, \bar{u}_j \bar{v}_j^\top \tag{4}$$

The next parameter update is therefore

$$\mathbf{W}_t \leftarrow \sum_i \sigma_i u_i v_i^\top + \sum_j \bar{\sigma}\, \bar{u}_j \bar{v}_j^\top \tag{5}$$

In Muon, as both the weights and the updates have a higher effective rank than Adam, we hypothesize there is a higher probability for singular-vector pair $u_i v_i^\top$ to align with $\bar{u}_j \bar{v}_j^\top$. This could cause the corresponding singular value of $\mathbf{W}_t$ to increase additively.

Attention-specific amplification Attention logits are computed via the bilinear form

$$q_i \cdot k_j = (x_i \mathbf{W}_q) \cdot (x_j \mathbf{W}_k). \tag{6}$$

The product $\mathbf{W}_q \mathbf{W}_k^\top$ squares the spectral norm, so any singular-value increase in either matrix is compounded. Muon's tendency to enlarge singular values therefore translates into a higher risk of logit explosion.

F K2 Critic Rubrics for General RL

F.1 Core Rubrics

- **Clarity and Relevance:** Assesses the extent to which the response is succinct while fully addressing the user's intent. The focus is on eliminating unnecessary detail, staying aligned with the central query, and using efficient formats such as brief paragraphs or compact lists. Unless specifically required, long itemizations should be avoided. When a choice is expected, the response should clearly offer a single, well-defined answer.
- **Conversational Fluency and Engagement:** Evaluates the response's contribution to a natural, flowing dialogue that extends beyond simple question-answering. This includes maintaining coherence, showing appropriate engagement with the topic, offering relevant observations or insights, potentially guiding the conversation constructively when appropriate, using follow-up questions judiciously, handling hypothetical or personal-analogy queries gracefully, and adapting tone effectively to suit the conversational context (e.g., empathetic, formal, casual).
- **Objective and Grounded Interaction:** Assesses the response's ability to maintain an objective and grounded tone, focusing squarely on the substance of the user's request. It evaluates the avoidance of both metacommentary (analyzing the query's structure, topic combination, perceived oddity, or the nature of the interaction itself) and unwarranted flattery or excessive praise directed at the user or their input. Excellent responses interact respectfully but neutrally, prioritizing direct, task-focused assistance over commentary on the conversational dynamics or attempts to curry favor through compliments.

F.2 Prescriptive Rubrics

- **Initial Praise:** Responses must not begin with compliments directed at the user or the question (e.g., "That's a beautiful question", "Good question!").
- **Explicit Justification:** Any sentence or clause that explains why the response is good or how it successfully fulfilled the user's request. This is different from simply describing the content.

F.3 Limitations

One potential side effect of this evaluation framework is that it may favor responses that appear confident and assertive, even in contexts involving ambiguity or subjectivity. This stems from two key constraints in the current rubric:

- **Avoidance of Self-Qualification:** The prescriptive rules prohibit self-assessments, explicit disclaimers, or hedging language (e.g., "this may not be accurate", "I might be wrong"). While these phrases can reflect epistemic humility, they are often penalized as non-informative or performative.
- **Preference for Clarity and Singularity:** The rubric reward direct, decisive answers when users ask for a recommendation or explanation. In complex or open-ended scenarios, this may disincentivize appropriately cautious or multi-perspective responses.

我们将更新矩阵写作

$$\Delta \mathbf{W}_t = \sum_j \bar{\sigma}\, \bar{u}_j \bar{v}_j^\top \tag{4}$$

因此，下一个参数更新为

$$\mathbf{W}_t \leftarrow \sum_i \sigma_i u_i v_i^\top + \sum_j \bar{\sigma}\, \bar{u}_j \bar{v}_j^\top \tag{5}$$

在 Muon 中，由于权重和更新的有效秩均高于 Adam，我们假设奇异向量对 $u_i v_i^\top$ 与 $\bar{u}_j \bar{v}_j^\top$ 对齐的概率更高。这可能导致 $\mathbf{W}_t$ 对应的奇异值以加法方式增大。

注意力特定放大 注意力logits通过双线性形式计算

$$q_i \cdot k_j = (x_i \mathbf{W}_q) \cdot (x_j \mathbf{W}_k). \tag{6}$$

乘积 $\mathbf{W}_q \mathbf{W}_k^\top$ 会对谱范数进行平方，因此任一矩阵的奇异值增加都会被放大。Muon 倾向于放大奇异值，因此这转化为更高的 logit 爆炸风险。

F K2 通用强化学习批评量规

F.1 核心评分标准

- 清晰度与相关性：评估回答在简洁的同时能在多大程度上完全满足用户意图。重点在于剔除不必要的细节，紧扣核心问题，并采用高效格式，如简短段落或紧凑列表。除非特别需要，应避免冗长的分项列举。当需要做出选择时，回答应明确提供单一且定义清晰的答案。
- 对话流畅度与参与度：评估回复对自然、流畅对话的贡献，超越简单问答。包括保持连贯性、对话题展现恰当参与、提供相关观察或见解、在适当时机建设性地引导对话、审慎使用后续问题、优雅处理假设或个人类比查询，并根据对话情境有效调整语气（如共情、正式、随意）。
- 客观且务实的互动：评估回复保持客观、务实语调的能力，专注于用户请求的核心内容。它衡量是否避免了元评论（分析查询的结构、主题组合、感知到的奇怪之处或互动本身的性质）以及对用户或其输入的无端奉承或过度赞美。优秀的回复以尊重但中立的方式互动，优先提供直接、以任务为中心的协助，而非对对话动态的评论或通过赞美来讨好。

F.2 规定性评分标准

- 初始赞扬：回复不得以针对用户或问题的赞美开头（例如“这是个很棒的问题”“问得好！”）。
- 明确理由：任何解释回答为何优秀或如何成功满足用户请求的句子或从句。这与单纯描述内容不同。

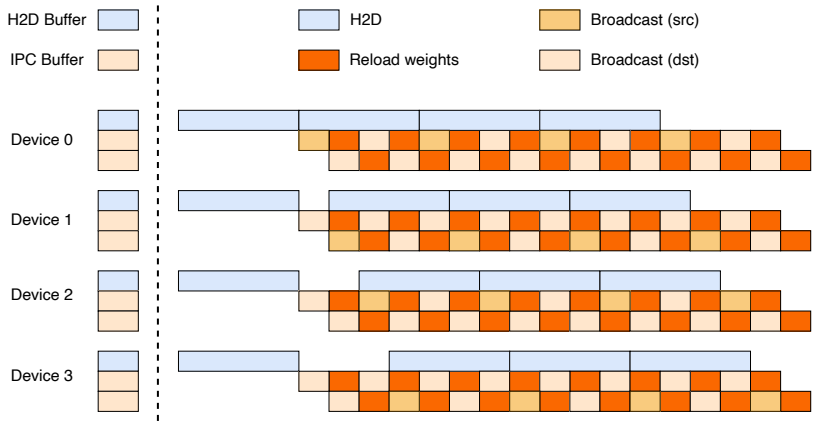
F.3 限制

这种评估框架的一个潜在副作用是，它可能倾向于那些显得自信且断言性强的回答，即便在涉及模糊或主观的情境中也是如此。这源于当前评分标准中的两个关键限制：

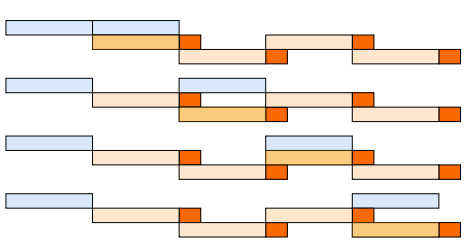
- 避免自我限定：规定性规则禁止自我评估、明确免责声明或模糊措辞（例如“这可能不准确”、“我可能是错的”）。尽管这些措辞可以体现认知谦逊，但它们常因被视为非信息性或表演性而受到惩罚。
- 偏好清晰与单一性：当用户寻求推荐或解释时，评分标准奖励直接、果断的回答。在复杂或开放式场景中，这可能会抑制适当谨慎或多视角的回答。

As a result, the model may occasionally overstate certainty in areas where ambiguity, nuance, or epistemic modesty would be more appropriate. Future iterations of the framework may incorporate more fine-grained handling of calibrated uncertainty.

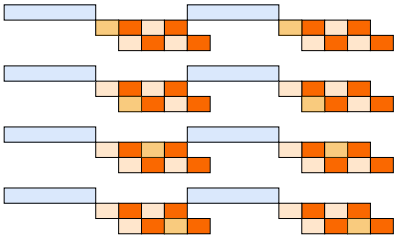
G Engine Switching Pipeline for RL Training



(a) Theoretical perfect three-stage pipeline weight update



(b) A PCIE bounded three-stage pipeline



(c) Fixed two-stage pipeline

Figure 13: pipeline for RL weight update

The *checkpoint engine* manages three equal-size device buffers on each GPU: an H2D buffer for loading the offloaded model parameters, and two IPC buffers for GPU-to-GPU broadcast. The IPC buffers are shared to inference engines, allowing it to directly access the same physical memory. These three buffers allow us to arrange the three steps in a pipeline.

Theoretical three-stage pipeline. As illustrated in Figure 13a, a three-stage pipeline is introduced. (1) *H2D*: a shard of the latest weights is copied into the H2D buffer asynchronously. (2) *Broadcast*: Once the copy completes, the shard will be copied to one IPC buffers and broadcast to all devices. (3) *Reload*: Inference engines simultaneously load parameters from the other IPC buffer.

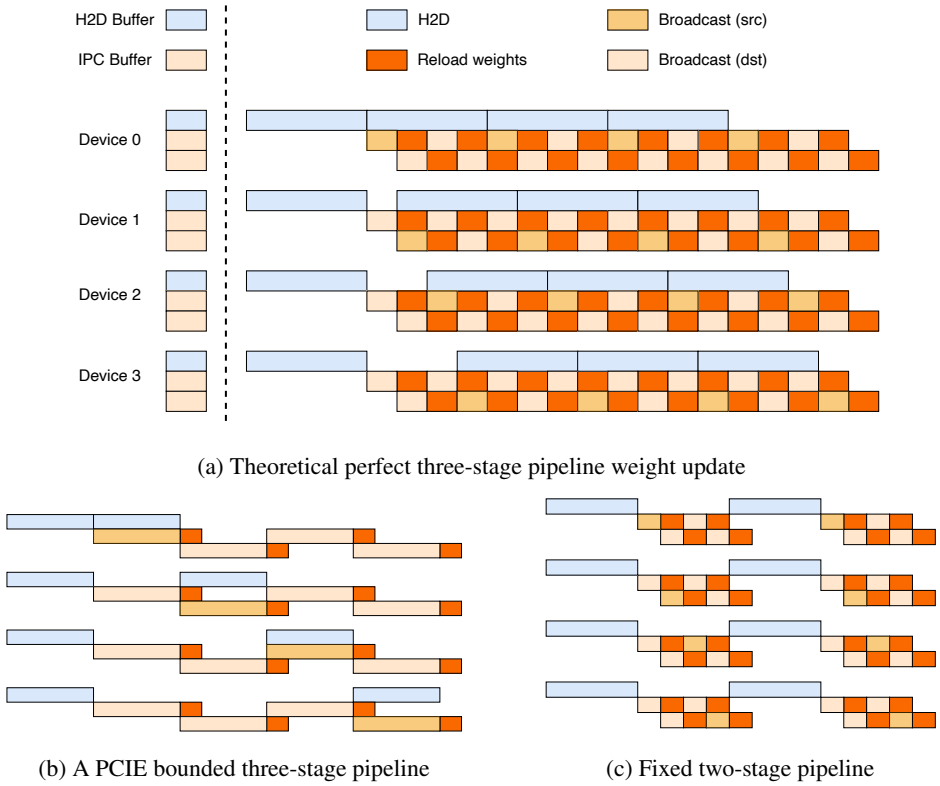
Two-stage pipeline due to PCIe saturation. On NVIDIA H800 clusters, concurrent H2D and broadcast saturate the shared PCIe fabric, collapsing the three stages into a sequential procedure (Figure 13b). We therefore adopt a simpler, two-stage scheme (Figure 13c): (1) All devices perform a single, synchronous H2D transfer. (2) The broadcast and reload proceed in parallel.

The two-stage pipeline will be bound by multiple synchronous H2D copy operations. But in large scale devices, model will be split into small shards, the entire parameter set fits into the H2D buffer in one transfer, the overhead will disappear.

By overlapping H2D, Broadcast, and Reload weights, we can obtain a high bandwidth to reshard the weights from train engines to all inference engines.

因此，该模型在某些情境下可能会过度表达确定性，而在这些情境中，模糊性、细微差别或认知谦逊可能更为恰当。框架的未来迭代可能会纳入更精细的校准不确定性处理方式。

用于强化学习训练的 G 引擎切换管线



(a) Theoretical perfect three-stage pipeline weight update

(b) A PCIE bounded three-stage pipeline

(c) Fixed two-stage pipeline

图13：RL权重更新流水线

checkpoint engine 在每个 GPU 上管理三个大小相等的设备缓冲区：一个用于加载已卸载模型参数的 H2D 缓冲区，以及两个用于 GPU 到 GPU 广播的 IPC 缓冲区。IPC 缓冲区被共享给推理引擎，使其能够直接访问同一段物理内存。这三个缓冲区使我们能够将这三个步骤安排成流水线。

理论三阶段流水线。如图13a所示，引入了一个三阶段流水线。(1) *H2D*：最新权重的分片被异步复制到H2D缓冲区。(2) *Broadcast*：复制完成后，该分片将被复制到一个IPC缓冲区并广播到所有设备。(3) *Reload*：推理引擎同时从另一个IPC缓冲区加载参数。

由于 PCIe 饱和而采用两阶段流水线。在 NVIDIA H800 集群上，并发的 H2D 和 broadcast 会饱和共享的 PCIe 结构，使三个阶段坍缩为顺序执行（图 13b）。因此我们采用更简单的两阶段方案（图 13c）：(1) 所有设备执行一次同步的 H2D 传输；(2) broadcast 与 reload 并行进行。

两阶段流水线将受到多个同步 H2D 拷贝操作的限制。但在大规模设备中，模型会被切分成小分片，整个参数集可以一次传输装入 H2D 缓冲区，开销将消失。

通过重叠 H2D、广播和重新加载权重，我们可以获得高带宽，将权重从训练引擎重新分片到所有推理引擎。