arXiv:2508.06471v1 [cs.CL] 8 Aug 2025

GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models

GLM-4.5 Team

Zhipu AI & Tsinghua University

(For the complete list of authors, please refer to the Contribution section)

Abstract

We present GLM-4.5, an open-source Mixture-of-Experts (MoE) large language model with 355B total parameters and 32B activated parameters, featuring a hybrid reasoning method that supports both thinking and direct response modes. Through multi-stage training on 23T tokens and comprehensive post-training with expert model iteration and reinforcement learning, GLM-4.5 achieves strong performance across agentic, reasoning, and coding (ARC) tasks, scoring 70.1% on TAU-Bench, 91.0% on AIME 24, and 64.2% on SWE-bench Verified. With much fewer parameters than several competitors, GLM-4.5 ranks 3rd overall among all evaluated models and 2nd on agentic benchmarks. We release both GLM-4.5 (355B parameters) and a compact version, GLM-4.5-Air (106B parameters), to advance research in reasoning and agentic AI systems. Code, models, and more information are available at `https://github.com/zai-org/GLM-4.5`.

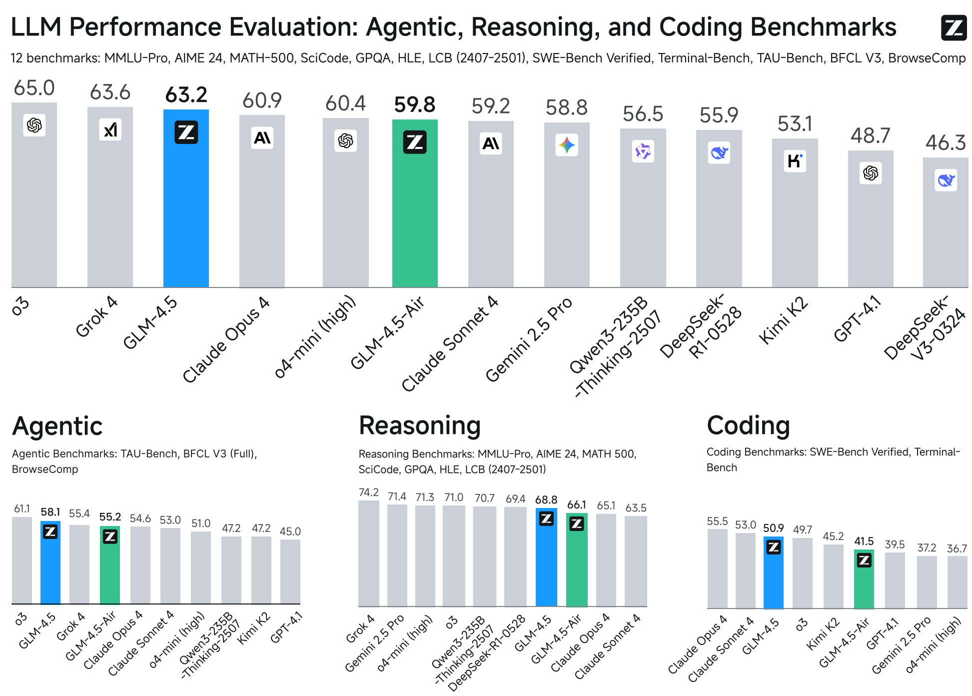

Figure 1: Average performance on agentic, reasoning, and coding (ARC) benchmarks. Overall, GLM-4.5 achieves a rank of 3rd, with GLM-4.5-Air following at rank 6th. The models listed are evaluated as of July 28, 2025.

GLM-4.5：智能体、推理与编程（ARC）基础模型

GLM-4.5 团队

智谱AI & 清华大学

(有关完整作者列表，请参阅贡献部分)

摘要

我们推出了GLM-4.5，一个开源的专家混合（MoE）大语言模型，总参数量为355B，激活参数量为32B，具有混合推理方法，支持思考模式和直接响应模式两种方式。通过在23T个token上进行多阶段训练，并结合专家模型迭代和强化学习的全面后训练，GLM-4.5在智能体、推理和编程（ARC）任务中表现出色，在TAU-Bench上得分70.1%，在AIME 24上得分91.0%，在SWE-bench Verified上得分64.2%。与多个竞争对手相比，GLM-4.5的参数量少得多，在所有评估的模型中总体排名第三，在智能体基准测试中排名第二。我们发布了GLM-4.5（355B参数）和精简版GLM-4.5-Air（106B参数），以推动推理和智能体AI系统的研究。代码、模型和更多信息可在https://github.com/zai-org/GLM-4.5获取。

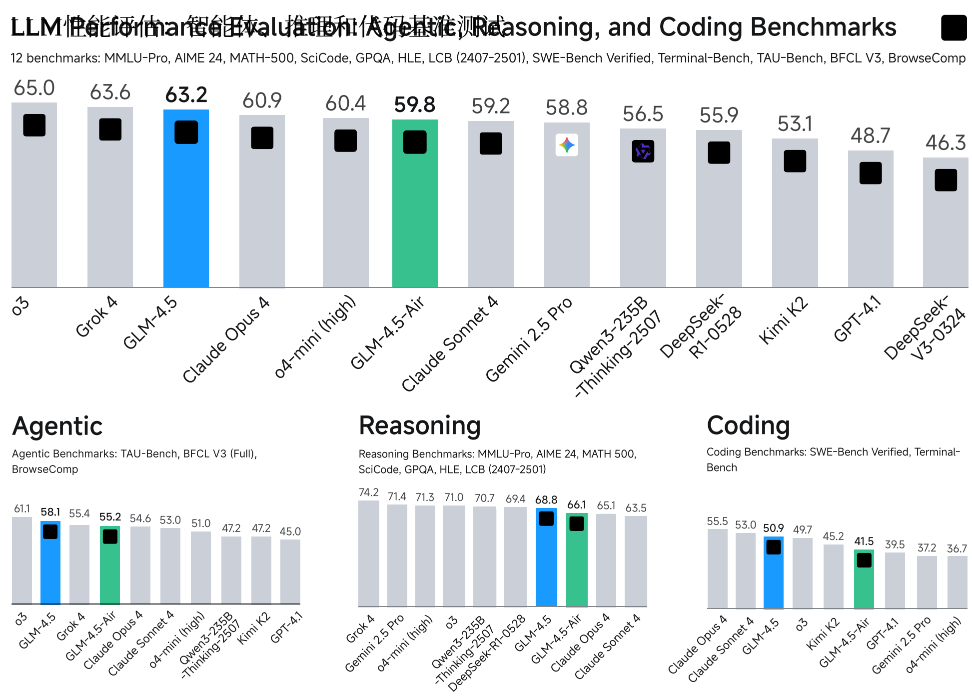

图1：在智能体、推理和编程（ARC）基准测试上的平均性能。总体而言，GLM-4.5排名第3，GLM-4.5-Air排名第6。所列模型的评估日期为2025年7月28日。

1 Introduction

Large language models (LLMs) are rapidly evolving from general knowledge repositories [6; 37; 50; 33; 23] into general problem-solvers. The ultimate ambition, often associated with Artificial General Intelligence (AGI), is to create models with human-level cognitive capabilities across diverse domains. This requires a unified mastery of complex problem-solving, generalization, and self-improvement, moving beyond task-specific excellence.

As LLMs become more integrated into real-world scenarios, the key to enhancing actual productivity and solving complex professional tasks lies in developing specific core capabilities. We identify three critical, interconnected capabilities as the measure of a truly generalist model: **Agentic** abilities for interacting with external tools and the real world; complex **Reasoning** for solving multi-step problems in domains like mathematics and science; and advanced **Coding** skills for tackling real-world software engineering tasks. While state-of-the-art proprietary models like OpenAI's o1/o3 [18] and Anthropic's Claude Sonnet 4 have demonstrated groundbreaking performance in specific ARC domains (e.g., mathematical reasoning or code fixing [20]), a single, powerful open-source model that excels across all three areas has remained elusive.

This paper introduces two new models: GLM-4.5 and GLM-4.5-Air, toward the goal of unifying all the different capabilities. The new models outperform existing open-source LLM models [13; 34; 47] across the board, with significant gains in agentic, reasoning, and coding tasks. GLM-4.5 and GLM-4.5-Air both feature hybrid reasoning modes: thinking mode for complex reasoning and agentic tasks, and non-thinking mode for instant responses. GLM-4.5 is our first MoE model, with 355B total parameters and 32B activated parameters. GLM-4.5 demonstrates strong performance on the following ARC benchmarks:

- **Agentic:** GLM-4.5 scores 70.1% on TAU-Bench and 77.8% on BFCL v3 [26], on par with Claude Sonnet 4. For web browsing agents, GLM-4.5 scores 26.4% on BrowseComp [45], clearly outperforming Claude Opus 4 (18.8%) and close to o4-mini-high (28.3%).
- **Reasoning:** GLM-4.5 demonstrates outstanding performance on a suite of challenging reasoning benchmarks, achieving 91.0% on AIME 24, 79.1% on GPQA [30], 72.9% on LiveCodeBench (2407-2501) [19], and 14.4% on HLE (Humanity's Last Exam) [28].
- **Coding:** GLM-4.5 scores 64.2% on SWE-bench Verified [20] and 37.5% on Terminal-Bench [35], outperforming GPT-4.1 and Gemini-2.5-pro, close to Claude Sonnet 4.

GLM-4.5-Air is a smaller MoE model with 106B parameters. It represents a significant leap among models at the 100B scale, matching or exceeding Qwen3-235B-A22B [47] and MiniMax-M1 [7].

In Figure 1, we show the average performance on 12 benchmarks across agentic, reasoning, and coding (ARC) tasks. Overall, GLM-4.5 is ranked in the **3rd** place and GLM-4.5-Air is ranked in the **6th**. On agentic tasks, GLM-4.5 is ranked in the 2nd place, following OpenAI o3. On coding tasks, GLM-4.5 is ranked in the third place, close to Claude Sonnet 4. Note that GLM-4.5 is highly parameter-efficient, with only half the parameters of DeepSeek-R1 [13] and one-third those of Kimi K2 [34]. In Figure 2, we report the scores on SWE-bench Verified vs model parameters of different open-source models, where GLM-4.5 and GLM-4.5-Air lie on the Pareto Frontier. More evaluation results are detailed in Section 4.

Both GLM-4.5 and GLM-4.5-Air are available on `Z.ai`, `BigModel.cn`, and also as open-source models on `https://huggingface.co/zai-org/GLM-4.5`. We also open-sourced an evaluation toolkit at `https://github.com/zai-org/glm-simple-evals` to ensure the reproducibility of our benchmark results.

2 Pre-Training

2.1 Architecture

In the GLM-4.5 series, we adopt the MoE architecture, which improves the computational efficiency of both training and inference. We employ loss-free balance routing [40] and sigmoid gates for MoE layers [23]. Different from DeepSeek-V3 [23] and Kimi K2 [34], we reduce the width (hidden dimension and number of routed experts) of the model and increase its height (number of layers), as we found that deeper models exhibited better reasoning capacity. In the self-attention component,

1 引言

大型语言模型（LLMs）正迅速从通用知识库[6; 37; 50; 33; 23]演变为通用问题解决者。这一终极目标，通常与通用人工智能（AGI）相关联，是创建在多个领域具备人类水平认知能力的模型。这需要对复杂问题解决、泛化和自我改进的统一掌握，超越特定任务的卓越表现。

随着LLMs越来越深入地融入现实场景，提高实际生产力并解决复杂专业任务的关键在于发展特定的核心能力。我们确定了三个关键且相互关联的能力，作为衡量真正通用模型的标准：与外部工具和现实世界交互的代理能力；解决数学和科学等领域多步骤问题的复杂推理能力；以及处理现实世界软件工程任务的高级编码技能。虽然像OpenAI的o1/o3 [18]和Anthropic的Claude Sonnet 4这样的最先进专有模型在特定ARC领域（例如数学推理或代码修复[20]）已经展示了突破性的性能，但一个在所有三个领域都表现出色的强大开源模型仍然难以实现。

本文介绍了两个新模型：GLM-4.5 和 GLM-4.5-Air，旨在统一所有不同的能力。这些新模型在所有方面都超越了现有的开源大语言模型 [13; 34; 47]，在代理、推理和编码任务方面取得了显著提升。GLM-4.5 和 GLM-4.5-Air 都具有混合推理模式：用于复杂推理和代理任务的思考模式，以及用于即时响应的非思考模式。GLM-4.5 是我们的第一个 MoE 模型，总参数量为 355B，激活参数量为 32B。GLM-4.5 在以下 ARC 基准测试中表现出强大的性能：

- 智能体：GLM-4.5在TAU-Bench上得分为70.1%，在BFCL v3上得分为77.8% [26]，与Claude Sonnet 4相当。对于网页浏览智能体，GLM-4.5在BrowseComp上得分为26.4% [45]，明显优于Claude Opus 4（18.8%），接近o4-mini-high（28.3%）。
- 推理能力：GLM-4.5 在一系列具有挑战性的推理基准测试中表现出色，在 AIME 24 上达到 91.0%，在 GPQA [30] 上达到 79.1%，在 LiveCodeBench (2407-2501) [19] 上达到 72.9%，在 HLE (Humanity's Last Exam) [28] 上达到 14.4%。
- 编程：GLM-4.5 在 SWE-bench Verified [20] 上得分为 64.2%，在 Terminal-Bench [35] 上得分为 37.5%，优于 GPT-4.1 和 Gemini-2.5-pro，接近 Claude Sonnet 4。

GLM-4.5-Air 是一个拥有 106B 参数的较小 MoE 模型。它在 100B 规模的模型中代表了显著的飞跃，匹配或超过了 Qwen3-235B-A22B [47] 和 MiniMax-M1 [7]。

在图1中，我们展示了在代理、推理和编码(ARC)任务上12个基准测试的平均性能。总体而言，GLM-4.5排名第3，GLM-4.5-Air排名第6。在代理任务上，GLM-4.5排名第2，仅次于OpenAI o3。在编码任务上，GLM-4.5排名第3，接近Claude Sonnet 4。请注意，GLM-4.5具有很高的参数效率，参数量仅为DeepSeek-R1 [13]的一半，是Kimi K2 [34]的三分之一。在图2中，我们报告了不同开源模型在SWE-bench Verified上的分数与模型参数的关系，其中GLM-4.5和GLM-4.5-Air位于帕累托前沿。更多评估结果详见第4节。

GLM-4.5 和 GLM-4.5-Air 都可以在 Z.ai 和 BigModel.cn 上使用，也可以作为开源模型在 https://huggingface.co/zai-org/GLM-4.5 上获取。我们还开源了一个评估工具包，位于 https://github.com/zai-org/glm-simple-evals，以确保我们基准测试结果的可复现性。

2 预训练

2.1 架构

在GLM-4.5系列中，我们采用了MoE架构，提高了训练和推理的计算效率。我们采用无损失平衡路由[40]和MoE层的sigmoid门[23]。与DeepSeek-V3 [23]和Kimi K2 [34]不同，我们减少了模型的宽度（隐藏维度和路由专家数量），增加了其高度（层数数量），因为我们发现更深层次的模型表现出更好的推理能力。在自注意力组件中，

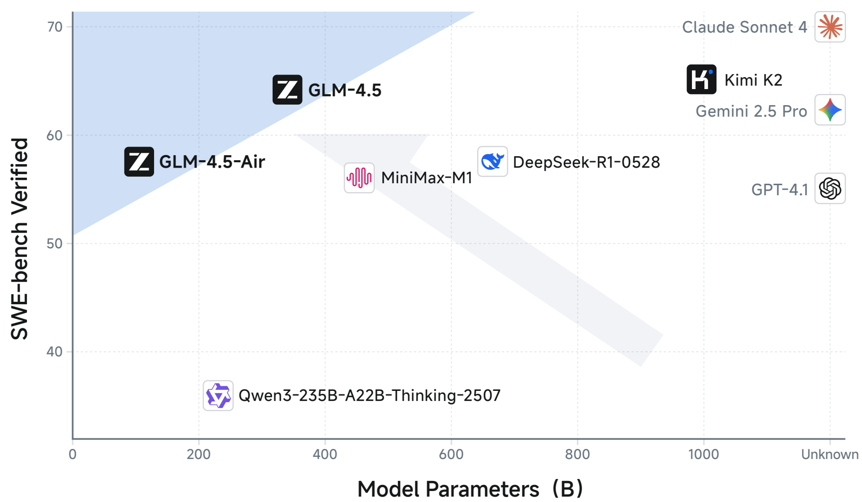

Figure 2: SWE-bench verified scores vs model parameters. Proprietary models are listed as unknown at the right side.

we employ Grouped-Query Attention with partial RoPE. Furthermore, we utilize 2.5 times more attention heads (96 heads for a 5120 hidden dimension). Counterintuitively, while this increased head count does not improve training loss compared to models with fewer heads, it consistently improves performance on reasoning benchmarks such as MMLU and BBH. We also incorporate QK-Norm [15] to stabilize the range of attention logits. For both GLM-4.5 and GLM-4.5-Air, we add an MoE layer as the MTP (Multi-Token Prediction) layer [12] to support speculative decoding during inference.

Table 1: Model architecture of GLM-4.5 and GLM-4.5-Air. When counting parameters, for GLM-4.5 and GLM-4.5-Air, we include the parameters of MTP layers but not word embeddings and the output layer.

Model	**GLM-4.5**	**GLM-4.5-Air**	**DeepSeek-V3**	**Kimi K2**
# Total Parameters	355B	106B	671B	1043B
# Activated Parameters	32B	12B	37B	32B
# Dense Layers	3	1	3	1
# MoE Layers	89	45	58	60
# MTP Layers	1	1	1	0
Hidden Dim	5120	4096	7168	7168
Dense Intermediate Dim	12288	10944	18432	18432
MoE Intermediate Dim	1536	1408	2048	2048
Attention Head Dim	128	128	192	192
# Attention Heads	96	96	128	64
# Key-Value Heads	8	8	128	64
# Experts (total)	160	128	256	384
# Experts Active Per Token	8	8	8	8
# Shared Experts	1	1	1	1
QK-Norm	Yes	No	No	No

2.2 Pre-Training Data

Our pre-training corpus includes documents from webpages, social media, books, papers, and code repositories. We carefully design the data processing pipelines for different sources.

Web The majority of our pre-training documents are English and Chinese webpages crawled from the Internet. Inspired by Nemotron-CC [32], we divide the crawled webpages into buckets of different quality scores. We up-sample documents from the bucket with higher quality scores

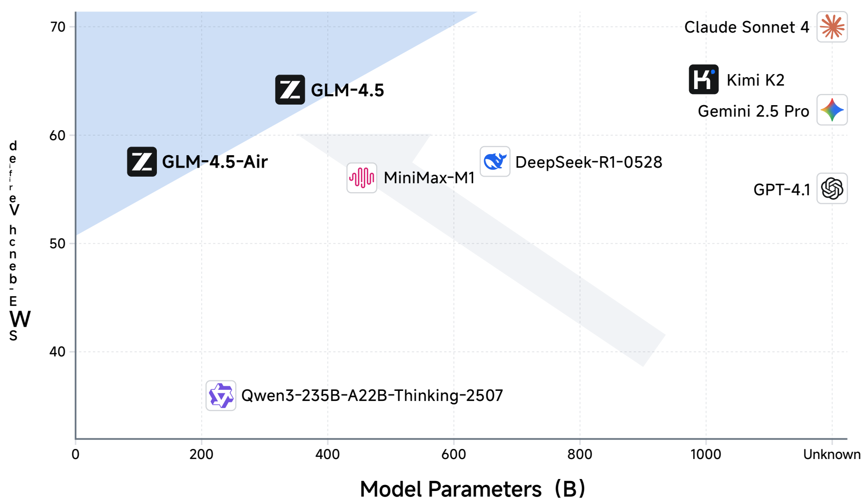

图2：SWE-bench验证分数与模型参数对比。专有模型在右侧列为未知。

我们采用带有部分RoPE的分组查询注意力（Grouped-Query Attention）。此外，我们使用了2.5倍更多的注意力头（对于5120的隐藏维度使用96个注意力头）。反直觉的是，虽然与头部数量较少的模型相比，这种增加的头部数量并不能改善训练损失，但它能持续提高在MMLU和BBH等推理基准上的性能。我们还集成了QK-Norm [15]来稳定注意力logits的范围。对于GLM-4.5和GLM-4.5-Air，我们都添加了一个MoE层作为MTP（多令牌预测）层[12]，以在推理过程中支持推测解码。

表1：GLM-4.5和GLM-4.5-Air的模型架构。在计算参数时，对于GLM-4.5和GLM-4.5-Air，我们包含MTP层的参数，但不包括词嵌入和输出层的参数。

Model	**GLM-4.5**	**GLM-4.5-Air**	**DeepSeek-V3**	**Kimi K2**
# Total Parameters	355B	106B	671B	1043B
# Activated Parameters	32B	12B	37B	32B
# Dense Layers	3	1	3	1
# MoE Layers	89	45	58	60
# MTP Layers	1	1	1	0
Hidden Dim	5120	4096	7168	7168
Dense Intermediate Dim	12288	10944	18432	18432
MoE Intermediate Dim	1536	1408	2048	2048
Attention Head Dim	128	128	192	192
# Attention Heads	96	96	128	64
# Key-Value Heads	8	8	128	64
# Experts (total)	160	128	256	384
# Experts Active Per Token	8	8	8	8
# Shared Experts	1	1	1	1
QK-Norm	Yes	No	No	No

2.2 预训练数据

我们的预训练语料库包括来自网页、社交媒体、书籍、论文和代码仓库的文档。我们为不同来源精心设计了数据处理管道。

网络 我们大部分的预训练文档是从互联网上抓取的英文和中文网页。受Nemotron-CC [32]的启发，我们将抓取的网页按不同的质量分数分成不同的桶。我们从质量分数较高的桶中对文档进行上采样

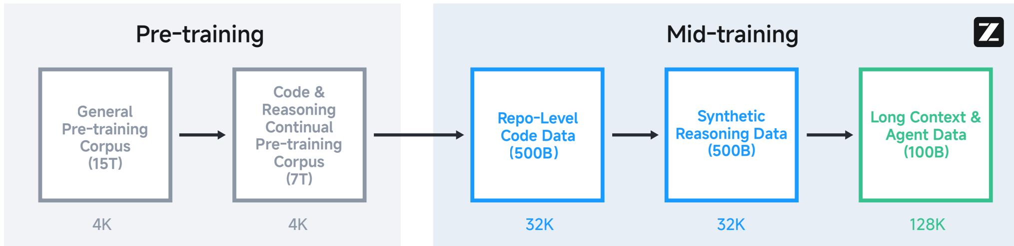

Figure 3: Pre-training and mid-training stages for GLM-4.5. We adapt a multi-stage training recipe and extend the sequence length from 4K to 128K.

and discard documents from the bucket with the lowest quality scores. The bucket with the highest quality scores contributes over 3.2 epochs during pre-training. In this way, the pre-training corpus can emphasize the high-frequency knowledge for reasoning tasks and also improve coverage for long-tail world knowledge. We have also found a large number of similar webpages automatically generated from templates and assigned high scores. Such webpages cannot be removed by MinHash deduplication. We additionally apply the SemDedup [1] pipeline to remove those similar webpages based on document embeddings.

Multilingual To support more natural languages, we include multilingual documents in our pre-training corpus. The multilingual corpus comes from both our crawled webpages and Fineweb-2 [27]. We apply a quality classifier that judges the educational utility of documents and up-sample high-quality multilingual documents.

Code We curated source code data from GitHub and various code hosting platforms. The code corpus undergoes a preliminary rule-based filtering, followed by classification using language-specific quality models that categorize samples into three tiers: high-quality, medium-quality, and low-quality. During training, we up-sampled high-quality code while excluding low-quality samples. Moreover, the Fill-In-the-Middle [5] training objective is applied to all source code data. For code-related web documents, we employ a two-stage retrieval process from our text pre-training corpus. Documents are initially selected based on two criteria: presence of HTML code tags, or identification by a FastText [22] classifier trained to detect code-related content. Subsequently, the retrieved documents undergo quality assessment using a dedicated model that classifies them into high-, medium-, or low-quality categories, following the same quality-based sampling strategy for source code. Finally, a fine-grained parser is employed to re-parse the selected web pages to better preserve the formats and contents of the code.

Math & Science To enhance the reasoning capacity, we collect documents related to mathematics and science from webpages, books, and papers. We apply a large language model to score candidate documents based on the ratio of educational content about mathematics and science, and train a small-scale classifier to predict the scores. Documents in the pre-training corpus with scores above a certain threshold are up-sampled.

The pre-training process of GLM-4.5 is divided into two stages. In the first stage, the model is mainly trained on general documents from webpages. During the second stage, we up-sample the source code from GitHub and webpages related to coding, mathematics, and science.

2.3 Mid-Training: Boost Reasoning & Agentic Capacity

After pre-training, we add several stages to further boost the model's performance on important application areas. Unlike traditional pre-training on large-scale general documents, these training stages utilize medium-size domain-specific datasets, including instruction data. Therefore, we denote these training stages as mid-training, which includes the following.

Repo-level Code Training At this training stage, we add concatenated code files from the same repository to learn cross-file dependency. To improve the model's software engineering capability,

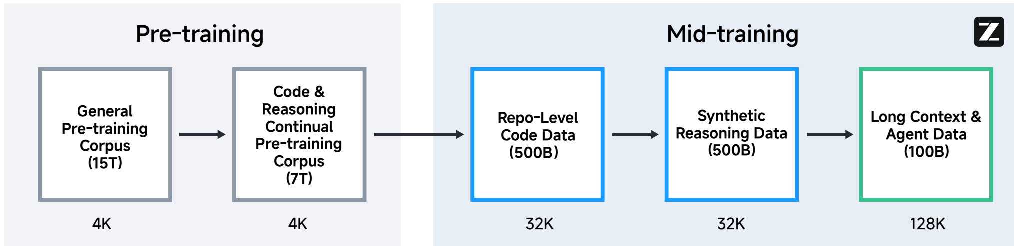

图3：GLM-4.5的预训练和中期训练阶段。我们采用多阶段训练方案，并将序列长度从4K扩展到128K。

并丢弃质量分数最低的桶中的文档。质量分数最高的桶在预训练过程中贡献了超过3.2个epoch。通过这种方式，预训练语料可以强调推理任务的高频知识，同时也能提高对长尾世界知识的覆盖范围。我们还发现大量由模板自动生成并被赋予高分的相似网页。这类网页无法通过MinHash去重来移除。我们额外应用了SemDedup [1]流程，基于文档嵌入来移除那些相似的网页。

多语言 为了支持更多的自然语言，我们在预训练语料库中包含了多语言文档。多语言语料库来自我们爬取的网页和Fineweb-2 [27]。我们应用了一个质量分类器，用于评估文档的教育效用，并对高质量的多语言文档进行上采样。

代码 我们从GitHub和各种代码托管平台收集了源代码数据。代码语料库经过初步的基于规则的过滤，然后使用特定语言的质量模型进行分类，将样本分为三个等级：高质量、中等质量和低质量。在训练过程中，我们对高质量代码进行上采样，同时排除低质量样本。此外，Fill-In-the-Middle [5]训练目标应用于所有源代码数据。对于与代码相关的网络文档，我们从文本预训练语料库中采用两阶段检索过程。文档最初根据两个标准进行选择：存在HTML代码标签，或由经过训练以检测代码相关内容的FastText [22]分类器识别。随后，检索到的文档使用专用模型进行质量评估，将其分为高质量、中等质量或低质量类别，遵循与源代码相同的质量基础采样策略。最后，采用细粒度解析器重新解析选定的网页，以更好地保留代码的格式和内容。

数学与科学 为了增强推理能力，我们从网页、书籍和论文中收集与数学和科学相关的文档。我们应用大型语言模型根据数学和科学教育内容的比例对候选文档进行评分，并训练一个小型分类器来预测这些分数。预训练语料库中分数超过特定阈值的文档会被上采样。

GLM-4.5的预训练过程分为两个阶段。在第一阶段，模型主要在来自网页的通用文档上进行训练。在第二阶段，我们对来自GitHub以及与编程、数学和科学相关的网页中的源代码进行上采样。

2.3 中期训练：提升推理能力与代理能力

在预训练之后，我们添加了几个阶段，以进一步提高模型在重要应用领域的性能。与传统的大规模通用文档预训练不同，这些训练阶段利用中等规模的领域特定数据集，包括指令数据。因此，我们将这些训练阶段称为中训练，包括以下内容。

仓库级代码训练 在这个训练阶段，我们将来自同一仓库的连接代码文件添加进来，以学习跨文件依赖关系。为了提高模型的软件工程能力，

we also include model-filtered issues, pull requests (PRs), and commits from GitHub, with related issues, PRs, and commits concatenated into one context and commits organized in a diff-like format. We extend the training sequence length from 4K to 32K to incorporate large repositories.

Synthetic Reasoning Data Training At this stage, we add synthetic reasoning content for math, science, and coding competitions. We collect a large number of questions and answers related to the reasoning tasks from webpages and books, and synthesize reasoning processes with a reasoning model.

Long-context & Agent Training To further boost the model's long-context performance, we extend the training sequence length from 32K to 128K and up-sample long documents from the pre-training corpus. Large-scale synthetic agent trajectories are also incorporated at this stage.

In Figure 3, we show the complete stages for pre-training and mid-training. The maximum sequence length is kept at 4,096 in pre-training, and is extended from 32,768 to 131,072 in mid-training. During pre-training, we did not use best-fit packing [11] since random truncation is a good data-augmentation strategy for pre-training documents. For datasets in mid-training, we applied best-fit packing to avoid truncating the reasoning process or repo-level code.

2.4 Hyper-Parameters

We employed the Muon optimizer [21; 24] for all parameters except word embedding, bias, and weights for RMSNorm. For hyperparameters, we set the Newton-Schulz iteration steps N to 5, momentum μ to 0.95, and scaled Muon's update RMS to 0.2. We observed that the Muon optimizer can accelerate convergence and tolerate larger batch sizes. We used cosine decay schedule for learning rate, instead of warmup-stable-decay (WSD) schedule [17]. Our early experiments showed that models trained with the WSD schedule perform worse on general benchmarks (SimpleQA, MMLU), indicating underfitting in the stable stage. The learning rate went through a warm-up stage from 0 to 2.5e-4 and a decaying stage to 2.5e-5 until the end of mid-training. We used a batch size warmup strategy, where the batch size was gradually increased from 16M tokens to 64M tokens in the training of the first 500B tokens, and remained constant in the remaining of training. For regularization, we set the weight decay ratio to 0.1 and did not use dropout. We set the maximum sequence length to 4,096 during pre-training, and extended it to 32,768 and 131,072 during the mid-training stage as shown in Figure 3. When extending the sequence length to 32K, we also adjusted RoPE's base frequency from 10,000 to 1,000,000 for better long-context modeling ability. For loss-free balance routing, we set the bias update rate to 0.001 for the first 15T tokens, and to 0.0 for the remaining tokens. We also applied auxiliary sequence-level balance loss with a 0.0001 weight to avoid extreme imbalance within any single sequence. The MTP loss weight λ was set to 0.3 for the first 15T tokens, and to 0.1 for the remaining tokens.

3 Post-Training: Expert Model Iteration

We divide the post-training process into two distinct stages. In stage 1 (*Expert Training*), we construct expert models specializing in three domains: Reasoning, Agent, and General chat. In stage 2 (*Unified Training*), we employ self-distillation techniques to integrate multiple experts, ultimately delivering a comprehensive model capable of generating responses through both deliberative reasoning and direct response modes.

3.1 Supervised Fine-Tuning

We perform Supervised Fine-Tuning (SFT) at the beginning of both Stage 1 (*Expert Training*) and Stage 2 (*Unified Training*). In the expert training stage, the primary role of SFT is to provide a cold start, empowering the model with basic chat, reasoning, and tool-use capabilities, which can then be further enhanced in subsequent expert RL training to achieve improved performance. In the unified training stage, the purpose of SFT is to distill the capabilities of different expert models into one hybrid reasoning generalist capable of handling different types of tasks.

我们还包括从GitHub获取的经过模型过滤的问题、拉取请求(PRs)和提交，将相关的问题、PRs和提交连接到一个上下文中，并将提交以类似差异(diff)的格式组织。我们将训练序列长度从4K扩展到32K，以纳入大型仓库。

合成推理数据训练 在此阶段，我们为数学、科学和编程竞赛添加合成推理内容。我们从网页和书籍中收集大量与推理任务相关的问题和答案，并使用推理模型合成推理过程。

长上下文与智能体训练　为进一步提升模型的长上下文性能，我们将训练序列长度32K扩展到128K，并对预训练语料库中的长文档进行上采样。在此阶段，我们还融入了大规模的合成智能体轨迹。

在图3中，我们展示了预训练和中间训练的完整阶段。在预训练中，最大序列长度保持在4,096，而在中间训练中，序列长度从32,768扩展到131,072。在预训练期间，我们没有使用最佳适配打包[11]，因为随机截断是预训练文档的一个很好的数据增强策略。对于中间训练的数据集，我们应用了最佳适配打包，以避免截断推理过程或仓库级别的代码。

2.4 超参数

我们使用了Muon优化器[21; 24]来优化除词嵌入、偏置和RMSNorm权重之外的所有参数。对于超参数，我们将牛顿-舒尔茨迭代步骤N设置为5，动量μ设置为0.95，并将Muon的更新RMS缩放为0.2。我们观察到Muon优化器可以加速收敛并容忍更大的批量大小。我们使用余弦衰减调度来调整学习率，而不是使用预热-稳定-衰减(WSD)调度[17]。我们早期的实验表明，使用WSD调度训练的模型在通用基准测试(SimpleQA, MMLU)上表现更差，表明在稳定阶段存在欠拟合问题。学习率经历了从0到2.5e-4的预热阶段，以及衰减到2.5e-5的阶段，直到中期训练结束。我们使用了批量大小预热策略，在前500B个token的训练中，批量大小从16M token逐渐增加到64M token，并在剩余的训练中保持不变。对于正则化，我们将权重衰减比率设置为0.1，并且没有使用dropout。我们在预训练期间将最大序列长度设置为4,096，并在中期训练阶段将其扩展到32,768和131,072，如图3所示。当将序列长度扩展到32K时，我们还调整了RoPE的基础频率从10,000到1,000,000，以获得更好的长上下文建模能力。对于无损失平衡路由，我们将前15T个token的偏置更新率设置为0.001，其余token设置为0.0。我们还应用了权重为0.0001的辅助序列级平衡损失，以避免在任何单个序列中出现极端不平衡。MTP损失权重λ在前15T个token上设置为0.3，其余token上设置为0.1。

3 训练后：专家模型迭代

我们将后训练过程分为两个不同的阶段。在阶段1 (*Expert Training*) 中，我们构建了三个专业领域的专家模型：推理、代理和通用聊天。在阶段2 (*Unified Training*) 中，我们采用自蒸馏技术来整合多个专家，最终交付一个能够通过深思熟虑推理和直接响应模式生成响应的综合模型。

3.1 监督微调

我们在第一阶段(*Expert Training*)和第二阶段(*Unified Training*)的开始都进行了监督微调(SFT)。在专家训练阶段，SFT的主要作用是提供冷启动，赋予模型基本的聊天、推理和工具使用能力，这些能力随后可以在后续的专家强化学习训练中得到进一步增强，从而实现性能的提升。在统一训练阶段，SFT的目的是将不同专家模型的能力提炼成一个能够处理不同类型任务的混合推理通才。

Cold Start SFT During the cold-start phase, we utilize a small set of supervised fine-tuning (SFT) data with extended Chain-of-Thought (CoT) responses. This approach ensures that each expert model possesses adequate foundational ability prior to the reinforcement learning phase.

Overall SFT In the Overall SFT stage, we collect millions of samples covering reasoning tasks (math, code, science, etc.), general chat (writing, translation, summarization, chit chat, etc.), agentic tasks (basic tool using, coding ability especially for authentic project development, etc.), and long-context understanding tasks from the previously trained expert models, and train the base model with a maximum context length of 128K tokens. By distilling from the output of distinct experts, the model learns to apply the most effective long CoT reasoning for each task to arrive at accurate answers. Especially, recognizing that a prolonged thinking process is unnecessary for certain domains that demand quick responses (such as chit chat), we meticulously balanced training data containing full reasoning with data lacking explicit thought processes. This approach allows the model to operate in both the reflective and immediate response modes, thereby creating a *hybrid reasoning model*. Moreover, we find the following strategy helpful in preparing SFT data to derive optimal performance.

Reducing Character Escaping in Function Call Templates Although function call parameters are predominantly represented in JSON format in contemporary implementations, a significant challenge emerges when these parameters contain code segments. In such cases, a substantial proportion of characters within the code require escaping, compelling the model to generate extensive escape characters, thereby increasing the learning burden for the model. While this issue poses minimal concern for models primarily designed for general chat, it represents a non-trivial challenge for agentic foundation models where function calling is a core capability. To mitigate this limitation, we propose a novel function call template that encapsulates function call keys and values within XML-like special token tags. This approach substantially reduces the necessity for character escaping in code segments, as the vast majority of code can be represented in its native form without escaping. Experimental results demonstrate that the proposed function call template does not compromise the performance of function call execution while recucing escaping. The following example (Figure 4) illustrates the structure of our proposed function call template. Detailed code implementation can be found in our open-source repository.

```
<|system|>
# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"name": "get_weather", "description": "Get the weather of a city for a specific date.", "parameters": {"
type": "object", "properties": {"city": {"type": "string", "description": "The city to get weather for,
in Chinese."}, "date": {"type": "string", "description": "The date in YYYY-MM-DD format."}}, "required":
["city"]}}
</tools>

For each function call, output the function name and arguments within the following XML format:
<tool_call>{function-name}
<arg_key>{arg-key-1}</arg_key>
<arg_value>{arg-value-1}</arg_value>
<arg_key>{arg-key-2}</arg_key>
<arg_value>{arg-value-2}</arg_value>
...
</tool_call><|system|>
You are a helpful assistant.<|user|>
Today is June 26, 2024. Could you please check the weather in Beijing and Shanghai for tomorrow<|
assistant|>
<think>The user wants to check the weather of Beijing and Shanghai tomorrow. I need to call the
get_weather function respectively to check Beijing and Shanghai.</think>
I will call the get_weather function to check the weather in Beijing and Shanghai.
<tool_call>get_weather
<arg_key>city</arg_key>
<arg_value>Beijing</arg_value>
<arg_key>date</arg_key>
<arg_value>2024-06-27</arg_value>
</tool_call>
<tool_call>get_weather
<arg_key>city</arg_key>
<arg_value>Shanghai</arg_value>
<arg_key>date</arg_key>
```

冷启动SFT 在冷启动阶段，我们利用一组带有扩展思维链（CoT）响应的小型监督微调（SFT）数据。这种方法确保每个专家模型在强化学习阶段之前具备足够的基础能力。

在整体SFT阶段，我们从先前训练的专家模型中收集数百万个样本，涵盖推理任务（数学、代码、科学等）、通用聊天（写作、翻译、摘要、闲聊等）、代理任务（基本工具使用、编码能力，特别是真实项目开发等）以及长上下文理解任务，并使用最大上下文长度为128K个令牌来训练基础模型。通过从不同专家的输出中提炼，模型学会了为每个任务应用最有效的长思维链推理，以得出准确的答案。特别是，我们认识到对于需要快速响应的某些领域（如闲聊），冗长的思考过程是不必要的，因此我们精心平衡了包含完整推理的数据与缺乏明确思维过程的数据。这种方法使模型能够以反思式和即时响应两种模式运行，从而创建了一个*hybrid reasoning model*。此外，我们发现以下策略在准备SFT数据以获得最佳性能方面很有帮助。

减少函数调用模板中的字符转义 尽管在当代实现中，函数调用参数主要以JSON格式表示，但当这些参数包含代码段时，会出现一个重大挑战。在这种情况下，代码中大部分字符需要转义，迫使模型生成大量转义字符，从而增加了模型的学习负担。虽然这个问题对于主要为通用聊天设计的模型来说几乎没有影响，但对于函数调用是核心能力的智能基础模型来说，这是一个不容忽视的挑战。为了缓解这一限制，我们提出了一种新颖的函数调用模板，该模板将函数调用键和值封装在类似XML的特殊标记标签中。这种方法大大减少了代码段中字符转义的需要，因为绝大多数代码可以以其原生形式表示，无需转义。实验结果表明，所提出的函数调用模板在减少转义的同时，不会损害函数调用执行的性能。以下示例（图4）说明了我们提出的函数调用模板的结构。详细的代码实现可以在我们的开源仓库中找到。

```
<|system|> # 工具 您可以调用一个或多个函数来协助用户查询。

您提供了函数签名在 <tools></tools> XML 标签中：<tools>

{"name": "get_weather", "description": "获取特定日期的城市天气。", "parameters": {"type": "object", "properties": {"city": {"type": "string", "description": "要获取天气的城市，使用中文。"}, "date": {"type": "string", "description": "日期格式为YYYY-MM-DD。"}}, "required": ["city"]}}
</工具>

对于每个函数调用，输出函数名和参数，使用以下XML格式：<tool_call>{function-name} <arg_key>{arg-key-1}</arg_key>arg_value>{arg-value-1}</arg_value>arg_key>{arg-key-2}</arg_key>arg_value>{arg-value-2}</arg_value>

... </tool_call><|system|> 你是一个有用的助手。<|user|> 今天是2024年6月26日。请帮我查询北京和上海明天的天气<|assistant|>think>用户想查询北京和上海明天的天气。我需要分别调用get_weather函数来查询北京和上海的天气。</think> 我将调用get_weather函数来查询北京和上海的天气。 <tool_call>get_weather <arg_key>city</arg_key>arg_value>北京</arg_value>arg_key>date</arg_key>arg_value>2024-06-27</arg_value>/tool_call>tool_call>get_weather <arg_key>city</arg_key>arg_value>上海</arg_value>arg_key>date</arg_key>
```

```
<arg_value>2024-06-27</arg_value>
</tool_call><|observation|>
<tool_response>
{"city": "Beijing", "date": "2024-06-27", "weather": "Sunny", "temperature": "26C"}
</tool_response>
<tool_response>
{"city": "Shanghai", "date": "2024-06-27", "weather": "Overcast", "temperature": "29C"}
</tool_response><|assistant|>
<think>I have obtained the weather query results of get_weather for Beijing and Shanghai respectively and
 can reply to users directly.</think>
It will be sunny in Beijing tomorrow with a temperature of 26 degrees Celsius. The weather in Shanghai is
 overcast with a temperature of 29 degrees Celsius.<|user|>
```

Figure 4: One example of function call template.

Rejection Sampling When sampling from expert models, we employ a comprehensive multistage filtering pipeline that includes: (1) removing repetitive, excessively short, or truncated samples, as well as those that fail to conform to valid reasoning formats; (2) conducting correctness verification for samples with objective answers; (3) utilizing reward models to filter responses to subjective questions; and (4) for tool-calling scenarios, ensuring adherence to proper tool invocation protocols and verification that trajectories reach the expected terminal states.

Prompt Selection and Response-Level Scaling Filtering challenging prompts and conducting response scaling on them prove to be effective. We experimented with removing the prompts in the bottom 50% based on response lengths, resulting in a 2%-4% improvement in math and science tasks, despite training with only half the data. Notably, we found that applying response scaling to these hard prompts can lead to further gains. Generating four responses for each prompt brought an additional 1%-2% improvement.

Automatic Agentic SFT Data Construction The construction of agentic SFT data involves four steps: 1. *Agentic Framework and Tool Collection*: We gather a set of agentic frameworks and real-world tool APIs and MCP servers, while also leveraging LLMs to automatically construct and simulate a batch of tools. 2. *Task Synthesis*: Based on these frameworks and tools, we automatically synthesize a collection of agentic tasks. On the one hand, for relatively mature frameworks, we leverage LLMs to comprehend their functionalities and automatically generate relevant queries or tasks. On the other hand, for more fragmented or disparate tools, we first select a representative subset and similarly employ LLMs to construct tasks about this subset. These tasks encompass both single-step and multi-step tool calling scenarios. 3. *Trajectory Generation*: For each synthesized task, we utilize existing LLMs to generate tool-call trajectories. Additionally, by employing the LLM as a user simulator, multi-step tool-call tasks are converted into trajectories involving multiple rounds of dialogue. 4. *Quality Filtering*: For each trajectory, multiple judge agents are used to evaluate whether the task is completed. Only successful trajectories are retained.

3.2 Reasoning RL

Reasoning RL focuses on enhancing a model's capabilities in domains that demand logical deduction, structured problem-solving, and verifiable accuracy. This includes critical areas such as mathematics, code generation, and scientific reasoning. A defining characteristic of these tasks is the high precision of their reward signals, as correctness can often be determined programmatically or with objective clarity. Mastery in these areas is not only crucial for advancing the raw intelligence of models but also serves as a fundamental building block for more complex, multi-step agentic behaviors. Recognizing the unique challenges and opportunities within reasoning RL, we have developed a suite of specialized techniques to effectively train our models. These methods, detailed below, are designed to address issues such as training efficiency, sample diversity, and data quality. Our overall RL algorithm builds upon the GRPO [31] framework, excluding the KL loss term. The comparison curves shown in this section are based on our smaller experimental model, not on GLM-4.5.

Difficulty-based Curriculum Learning During reinforcement learning, the model's proficiency evolves, creating a mismatch with static training data. In the later stages, as the model becomes more capable, overly simple data can lead to rollouts where all rewards are 1s. Conversely, in the early stages, excessively difficult data often results in batches where all rewards are 0s. In both

```
<
<arg_value>2024-06-27</arg_value>/tool_call><|observation|>tool_response> {"city": "Beijing", "date": "2024-06-27", "weather": "Sunny", "temperature": "26C"} </tool_response>tool_response> {"city": "Shanghai", "date": "2024-06-27", "weather": "Overcast", "temperature": "29C"} </tool_response><|assistant|>think>我已经分别获取了北京和上海的get_weather天气查询结果，可以直接回复用户。</think> 明天北京将是晴天，气温26摄氏度。上海天气阴天，气温29摄氏度。<|user|>
```

图4：函数调用模板的一个示例。

拒绝采样　当从专家模型进行采样时，我们采用了一个全面的多阶段过滤流程，包括(1) 移除重复、过短或不完整的样本，以及不符合有效推理格式的样本；(2) 对具有客观答案的样本进行正确性验证；(3) 利用奖励模型过滤主观问题的回答；(4) 对于工具调用场景，确保遵循适当的工具调用协议，并验证轨迹是否达到预期的终止状态。

提示选择和响应级别缩放：筛选具有挑战性的提示并对它们进行响应缩放被证明是有效的。我们尝试移除基于响应长度排在后50%的提示，尽管只使用了半数数据进行训练，但在数学和科学任务上仍取得了2%-4%的改进。值得注意的是，我们发现对这些困难提示应用响应缩放可以带来进一步的提升。为每个提示生成四个响应带来了额外的1%-2%的改进。

智能体SFT数据自动构建　智能体SFT数据的构建包含四个步骤：1. *Agentic Framework and Tool Collection*: 我们收集一组智能体框架和真实世界的工具API及MCP服务器，同时利用LLMs自动构建和模拟一批工具。2. *Task Synthesis*: 基于这些框架和工具，我们自动合成一组智能体任务。一方面，对于相对成熟的框架，我们利用LLMs理解其功能并自动生成相关查询或任务。另一方面，对于更零散或不同的工具，我们首先选择一个代表性子集，并同样使用LLMs构建关于此子集的任务。这些任务包括单步和多步工具调用场景。3. *Trajectory Generation*: 对于每个合成的任务，我们利用现有的LLMs生成工具调用轨迹。此外，通过将LLM用作用户模拟器，多步工具调用任务被转换为涉及多轮对话的轨迹。4. *Quality Filtering*: 对于每个轨迹，使用多个判断智能体评估任务是否完成。只有成功的轨迹被保留。

3.2 推理RL

推理强化学习专注于增强模型在需要逻辑推理、结构化问题解决和可验证准确性方面的能力。这包括数学、代码生成和科学推理等关键领域。这些任务的一个显著特征是其奖励信号的高精度，因为正确性通常可以通过编程或客观清晰度来确定。在这些领域的掌握不仅对提升模型的原始智能至关重要，而且为更复杂的多步骤智能体行为提供了基础构建模块。认识到推理强化学习中的独特挑战和机遇，我们开发了一系列专门技术来有效训练我们的模型。这些方法将在下面详细说明，旨在解决训练效率、样本多样性和数据质量等问题。我们的整体强化学习算法基于GRPO [31]框架构建，但排除了KL损失项。本节中显示的比较曲线基于我们较小的实验模型，而非GLM-4.5。

基于难度的课程学习 在强化学习过程中，模型的熟练程度不断演变，与静态训练数据产生不匹配。在后期阶段，随着模型能力的提升，过于简单的数据可能导致所有奖励都为1的展开结果。相反，在早期阶段，过于困难的数据常常导致所有奖励都为0的批次。在这两种

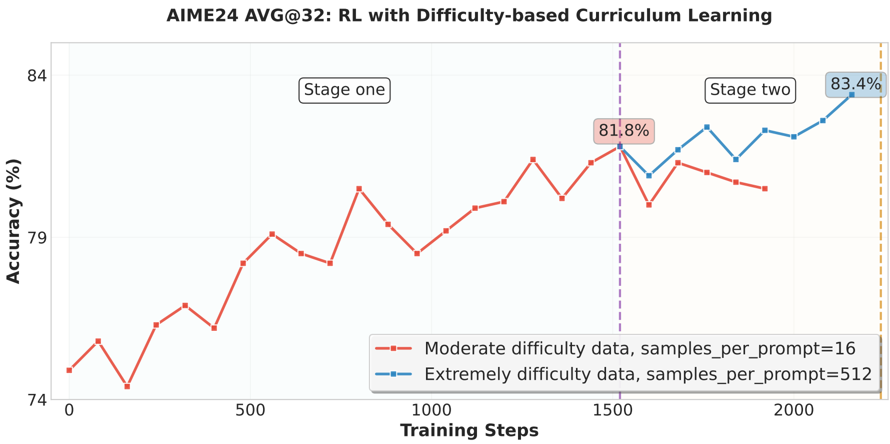

Figure 5: **Effectiveness of the two-stage difficulty-based curriculum on AIME'24.** The blue line (our method) switches to extremely difficult problems (pass@8=0, pass@512>0) in the second stage, showing continued improvement. The red line (baseline) continues with moderate-difficulty problems and plateaus.

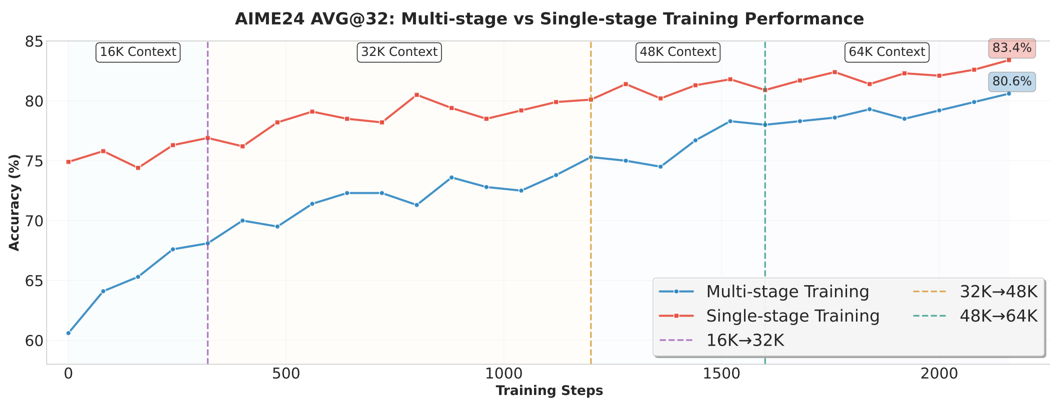

Figure 6: **Single-stage vs. multi-stage RL at 64K context length.** The red line (single-stage at 64K) achieves superior performance. The blue line (multi-stage with progressively increasing length) suffers from an irreversible performance drop in early stages, limiting its final performance.

scenarios, the lack of reward variance provides no useful gradient signal, severely hindering training efficiency. To address this challenge, we employ a two-stage difficulty-based curriculum for RL. The effectiveness of this and other strategies discussed below is validated through controlled experiments on a smaller model, which allows for rapid iteration and precise ablation studies. As shown in Figure 5, this two-stage approach enables the model to consistently surpass its performance ceiling. Crucially, to maintain high signal quality and reduce noise, all problems used in the second stage are strictly sourced from a pool with verified correct answers.

Single-Stage RL at 64K Output Length Previous research [25] has suggested conducting RL in multiple stages with progressively increasing maximum output lengths. However, our experiments reveal that this multi-stage approach is less effective than a single-stage RL process conducted directly at the maximum target length of 64K. Since the initial Supervised Fine-Tuning (SFT) has already conditioned the model on generating 64K-length responses, introducing RL stages with shorter maximum lengths can cause the model to "unlearn" its long-context capabilities. This often leads to a significant and irreversible drop in performance, as the model's average output length decreases. This degradation is difficult to recover from in the final 64K-length RL stage, thus limiting further improvement. Our experiments confirm this observation: as demonstrated in Figure 6, applying RL directly at the full 64K-length continually pushes the model's limits and yields better performance.

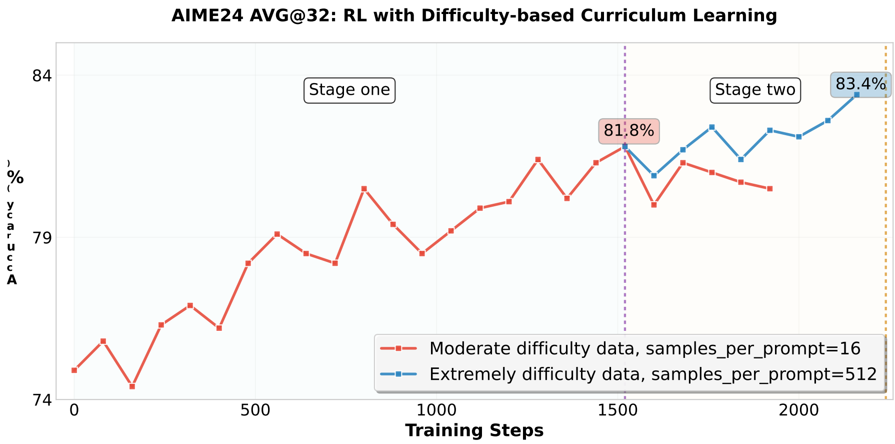

图5：基于难度的两阶段课程在AIME'24上的有效性。蓝线（我们的方法）在第二阶段切换到极难的问题（pass@8=0, pass@512>0），显示出持续改进。红线（基线）继续使用中等难度的问题并趋于平稳。

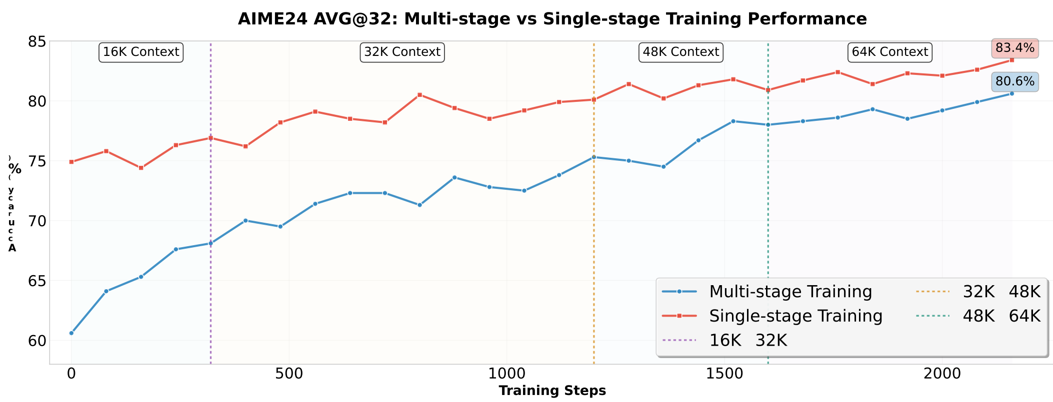

图6：64K上下文长度下单阶段与多阶段RL对比。红线（64K单阶段）实现了更优的性能。蓝线（渐进增加长度的多阶段）在早期阶段遭受了不可逆的性能下降，限制了其最终性能。

在场景中，奖励方差的缺乏无法提供有用的梯度信号，严重阻碍了训练效率。为应对这一挑战，我们采用了一种基于难度的两阶段课程进行强化学习(RL)。下面讨论的这种及其他策略的有效性通过在较小模型上的受控实验得到验证，这允许快速迭代和精确的消融研究。如图5所示，这种两阶段方法使模型能够持续超越其性能上限。关键的是，为了保持高信号质量和减少噪声，第二阶段使用的所有问题都严格来自具有已验证正确答案的题库。

单阶段64K输出长度的强化学习　先前的研究[25]建议在多个阶段进行强化学习，并逐步增加最大输出长度。然而，我们的实验表明，这种多阶段方法不如直接在64K的最大目标长度上进行单阶段强化学习过程有效。由于最初的监督微调(SFT)已经使模型习惯于生成64K长度的响应，引入具有较短最大长度的强化学习阶段可能导致模型"遗忘"其长上下文能力。这通常会导致性能的显著且不可逆转的下降，因为模型的平均输出长度减少了。这种退化在最终的64K长度强化学习阶段难以恢复，从而限制了进一步的改进。我们的实验证实了这一观察结果：如图6所示，在完整的64K长度上直接应用强化学习不断推动模型的极限，并产生更好的性能。

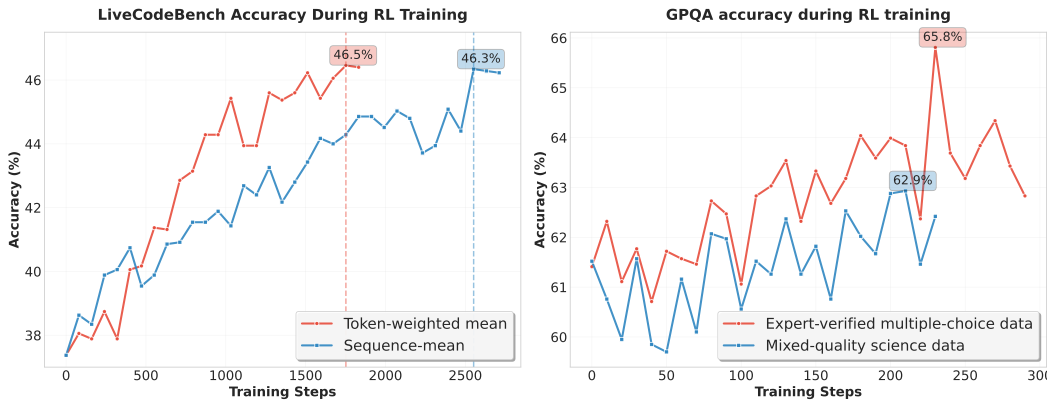

Figure 7: **Ablation Studies for Code and Science RL. (Left)** Comparison of loss calculation methods for code RL. The token-weighted mean loss approach achieves faster convergence compared to the sequence-mean loss baseline, accelerating the training process. **(Right)** Ablation on data sources for science RL on the GPQA-Diamond benchmark. Training exclusively on a small set of high-quality, expert-verified multiple-choice questions yields the best performance, significantly outperforming training on mixed-quality data.

Dynamic Sampling Temperature During RL, the sampling temperature is a key parameter for controlling trajectory diversity. A temperature that is too low leads to convergent, less exploratory outputs, while one that is too high introduces low-quality, noisy samples, undermining both model accuracy and training efficiency. Using a fixed sampling temperature is suboptimal because it fails to adapt as the policy distribution becomes more concentrated (i.e., has lower entropy), often resulting in insufficient exploration at later stages. Therefore, we propose dynamically adjusting the sampling temperature to maintain a healthy balance between accuracy and exploration. Specifically, when the average reward of rollouts stabilizes, we identify this as a convergence phase and increase the sampling temperature to encourage greater diversity. To mitigate the risk of introducing excessive noise, we implement a quality-control mechanism: we periodically evaluate model performance on a held-out validation set across a range of temperatures. The temperature for the next training phase is then set to the maximum value that does not cause a performance drop of more than 1% from the current optimum [2].

Code and Science RL Compared to mathematics, RL for coding and scientific domains has received less attention in the literature. We conducted extensive controlled RL experiments in these areas and arrived at the following empirical conclusions. For **code RL**, we find that the choice of loss calculation is critical for training efficiency. As illustrated in Figure 7 (left), adopting a token-weighted mean loss is highly beneficial compared to a conventional sequence-mean loss. The token-weighted approach provides a finer-grained and more stable gradient signal, which leads to significantly faster convergence. This method also helps to alleviate the length bias inherent in sequence-level rewards and effectively suppresses the generation of overly simplistic or repetitive "base case" samples during training. For **science RL**, our findings on the GPQA-Diamond benchmark highlight that data quality and type are paramount factors. As shown in Figure 7 (right), using exclusively expert-verified multiple-choice questions for RL leads to significantly better performance compared to training with mixed-quality or unverified data. This result underscores that even for tasks with simple formats like multiple-choice, rigorously filtering the RL data pool to include only high-quality, challenging instances is crucial for effective model improvement.

3.3 Agentic RL

Reinforcement Learning from Human Feedback (RLHF) helps language models follow human instructions more faithfully. Applying RL to math and programming contests has further uncovered strong reasoning abilities and favorable scaling behavior on tasks whose outcomes can be objectively verified. Building on these insights, we focus on agentic settings—specifically web-search and code-generation agents—where every action or answer can be automatically checked. This built-in verifiability supplies dense, reliable rewards, enabling us to scale RL training more effectively.

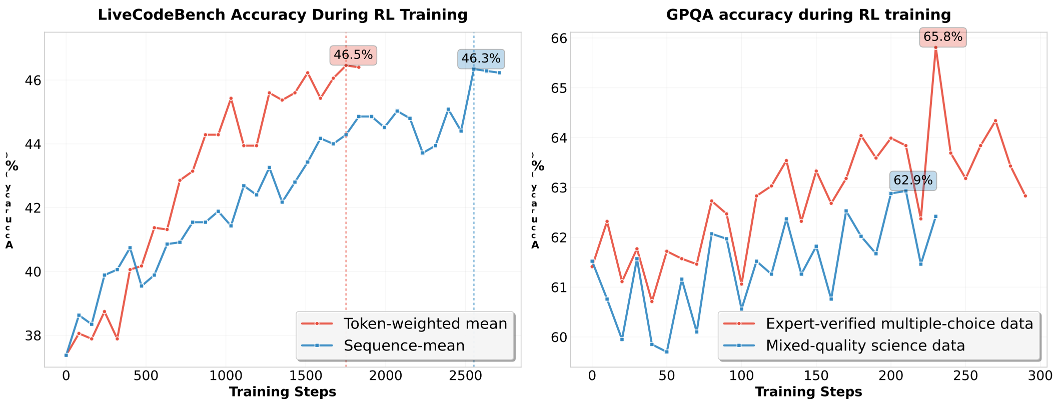

图7：代码和科学强化学习的消融研究。（左）代码强化学习中损失计算方法的比较。与序列平均损失基线相比，基于token加权的平均损失方法实现了更快的收敛速度，加速了训练过程。（右）在GPQA-Diamond基准上对科学强化学习的数据源进行消融研究。仅在一小组高质量、专家验证的多选题上进行训练能获得最佳性能，显著优于在混合质量数据上的训练。

动态采样温度 在强化学习(RL)过程中，采样温度是控制轨迹多样性的关键参数。温度过低会导致收敛性更强、探索性更弱的输出，而温度过高则会引入低质量的噪声样本，损害模型准确性和训练效率。使用固定的采样温度并非最优选择，因为它无法随着策略分布变得更加集中（即熵降低）而进行调整，常常导致后期探索不足。因此，我们提出动态调整采样温度，以在准确性和探索性之间保持健康平衡。具体而言，当回合平均奖励趋于稳定时，我们将其识别为收敛阶段，并提高采样温度以鼓励更大的多样性。为了降低引入过多噪声的风险，我们实施了一个质量控制机制：我们定期在保留的验证集上评估模型在不同温度下的性能。下一训练阶段的温度被设置为不会导致性能比当前最佳值下降超过1%的最大值[2]。

与数学相比，编程和科学领域的强化学习在文献中受到的关注较少。我们在这些领域进行了广泛的受控强化学习实验，并得出了以下经验性结论。对于代码强化学习，我们发现损失计算的选择对训练效率至关重要。如图7（左）所示，与传统的序列平均损失相比，采用标记加权平均损失非常有益。标记加权方法提供了更细粒度和更稳定的梯度信号，从而显著加快了收敛速度。这种方法还有助于缓解序列级奖励中固有的长度偏差，并有效抑制训练过程中生成过于简单或重复的"基本案例"样本。对于科学强化学习，我们在GPQA-Diamond基准测试上的发现突显出数据质量和类型是至关重要的因素。如图7（右）所示，与使用混合质量或未经验证的数据进行训练相比，仅使用专家验证的多项选择题进行强化学习会带来显著更好的性能。这一结果强调，即使对于像多项选择题这样格式简单的任务，严格筛选强化学习数据池以仅包含高质量、具有挑战性的实例，对于有效改进模型也至关重要。

3.3 智能体强化学习

基于人类反馈的强化学习(RLHF)帮助语言模型更忠实地遵循人类指令。将强化学习应用于数学和编程竞赛进一步发现了在结果可以被客观验证的任务上具有强大的推理能力和有利的扩展行为。基于这些见解，我们专注于代理设置——特别是网络搜索和代码生成代理——其中每个动作或答案都可以被自动检查。这种内置的可验证性提供了密集、可靠的奖励，使我们能够更有效地扩展强化学习训练。

3.3.1 Data Collection and Synthesis for Agents

For web-search tasks and open-domain information seeking, we develop a data-synthesis pipeline that yields demanding question–answer pairs requiring multi-step reasoning across multiple web sources. This corpus is designed to sharpen GLM's ability to uncover elusive, interwoven facts on the internet. Dataset construction blends two approaches: (1) an automated pipeline powered by multi-hop reasoning over knowledge graphs, and (2) human-in-the-loop extraction and selective obfuscation of content from several web pages to prepare reinforcement-learning training signals.

For software-engineering tasks, we curate an extensive collection of GitHub pull requests and issues to create a realistic software-development benchmark comprising user prompts and executable unit tests. All evaluations run inside a hardened sandbox with a distributed system, which provides both horizontal scalability and strong isolation guarantees.

3.3.2 Pushing the Limits with Reinforcement Learning and Iterative Self-distillation

We adopt the group-wise policy optimization algorithm for RL training. For each problem x, we sample K agent traces $\{y_1, \ldots, y_k\}$ from the previous policy π_{old}, and optimize the model π_θ with respect to the following objective:

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

where $\bar{r}(x) = \frac{1}{k}\sum_{i=1}^{k} r\big(x, y_i\big)$ is the mean reward of the sampled responses. It is noted that only model-generated tokens are used for optimization, and the environment feedback is ignored in loss computation.

Outcome Supervision with Process Action Format Penalty For web search tasks, we use the accuracy of the final answer as a reward for the entire agent trace. For coding agents, we primarily utilize SWE data with verifiable test cases for RL training. Our experiments have shown that RL training on web search and SWE tasks leads to generalized performance improvements across other tasks and benchmarks, such as general tool usage and coding tasks like Terminal-Bench. Additionally, we apply a process format penalty to ensure the model generates correct tool call formats. If the model fails to produce the correct tool format during agent trace generation, the process will be halted, and the trace will receive a zero reward.

Iterative Distillation Since RL training on agent tasks is time-consuming, we adopt a self-distillation approach to iteratively enhance the performance of the SFT cold-start model before resuming RL training on this improved model. Specifically, we first perform RL training on the initial cold-start model to boost agent performance. Once training has reached a certain step count or plateaued, we apply self-distillation by substituting the original cold-start data with responses generated by the RL-trained model, thus creating a superior SFT model. We then conduct further RL training on this enhanced model, progressively increasing training difficulty. This iterative strategy allows us to push the performance limits of RL-trained models efficiently.

Scaling Test-time Compute via Interaction Turns For agent tasks, we observe significant performance gains given increasing interaction turns with the environment. Compared to test-time scaling in reasoning models, which scales output tokens, agent tasks make use of test-time compute by continuously interacting with the environment, e.g., searching high and low for hard-to-find web information or writing test cases for self-verification and self-correction for coding tasks. Figure 8 shows that with varying browsing effort, accuracy scales smoothly with test-time compute.

3.4 General RL

General RL aims to holistically improve the model's overall performance, remediate potential issues, and strengthen key capabilities. Central to our methodology is a multi-source feedback system that synergizes rule-based feedback, human feedback (RLHF), and model-based feedback (RLAIF). This hybrid framework provides more robust training signals and allows us to leverage the unique advantages of each source: the precision of automated rules, the nuanced judgment of human annotators, and the scalability of AI-driven evaluation.

3.3.1 智能体的数据收集与合成

对于网络搜索任务和开放领域信息检索，我们开发了一个数据合成流水线，能够生成需要跨多个网络源进行多步推理的复杂问答对。该语料库旨在增强GLM在互联网上发现难以捉摸的交织事实的能力。数据集构建结合了两种方法：(1) 基于知识图谱多跳推理的自动化流水线，以及 (2) 人工参与从多个网页中提取内容并进行选择性模糊处理，以准备强化学习训练信号。

对于软件工程任务，我们整理了大量GitHub的拉取请求和问题，以创建一个包含用户提示和可执行单元测试的、真实的软件开发基准。所有评估都在一个加固的沙箱内运行，该沙箱配有分布式系统，既提供了水平扩展能力，又保证了强隔离性。

3.3.2 通过强化学习和迭代自蒸馏突破极限

我们采用分组策略优化算法进行强化学习训练。对于每个问题 x，我们从先前的策略 π_{old} 中采样 K 个智能体轨迹 $\{y_1, \dots, y_k\}$，并根据以下目标优化模型 π_θ：

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

其中 $\bar{r}(x) = \frac{1}{k}\sum_{i=1}^{k} r\big(x, y_i\big)$ 是采样响应的平均奖励。需要注意的是，仅使用模型生成的令牌进行优化，而在损失计算中忽略环境反馈。

结果监督与进程动作格式惩罚 对于网络搜索任务，我们使用最终答案的准确性作为整个代理轨迹的奖励。对于编码代理，我们主要利用具有可验证测试用例的SWE数据进行强化学习训练。我们的实验表明，在网络搜索和SWE任务上进行强化学习训练，能够带来在其他任务和基准测试上的泛化性能提升，例如通用工具使用和像Terminal-Bench这样的编码任务。此外，我们应用进程格式惩罚，以确保模型生成正确的工具调用格式。如果在代理轨迹生成过程中，模型未能产生正确的工具格式，则进程将被终止，并且该轨迹将获得零奖励。

迭代蒸馏 由于在代理任务上进行RL训练耗时较长，我们采用自蒸馏方法在恢复RL训练之前迭代增强SFT冷启动模型的性能。具体而言，我们首先在初始冷启动模型上进行RL训练以提高代理性能。一旦训练达到一定步数或趋于平稳，我们通过用RL训练模型生成的响应替换原始冷启动数据来应用自蒸馏，从而创建一个更优的SFT模型。然后，我们在这个增强的模型上进行进一步的RL训练，逐步提高训练难度。这种迭代策略使我们能够高效地推动RL训练模型的性能极限。

通过交互轮次扩展测试时计算 对于代理任务，我们观察到随着与环境交互轮次的增加，性能显著提升。与推理模型中的测试时扩展（扩展输出令牌）相比，代理任务通过持续与环境交互来利用测试时计算，例如，为难以找到的网络信息进行高搜索和低搜索，或在编码任务中编写测试用例进行自我验证和自我纠正。图8显示，随着浏览努力程度的变化，准确度随测试时计算平滑扩展。

3.4 通用强化学习

通用RL旨在全面提高模型的总体性能，解决潜在问题，并加强关键能力。我们方法的核心是一个多源反馈系统，它协同基于规则的反馈、人类反馈(RLHF)和基于模型的反馈(RLAIF)。这种混合框架提供了更强大的训练信号，使我们能够利用每种来源的独特优势：自动化规则的精确性、人工标注者的细致判断以及AI驱动评估的可扩展性。

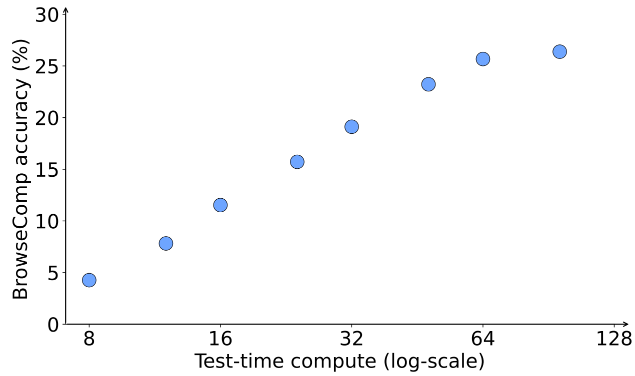

Figure 8: Interaction Turns Scaling for BrowseComp.

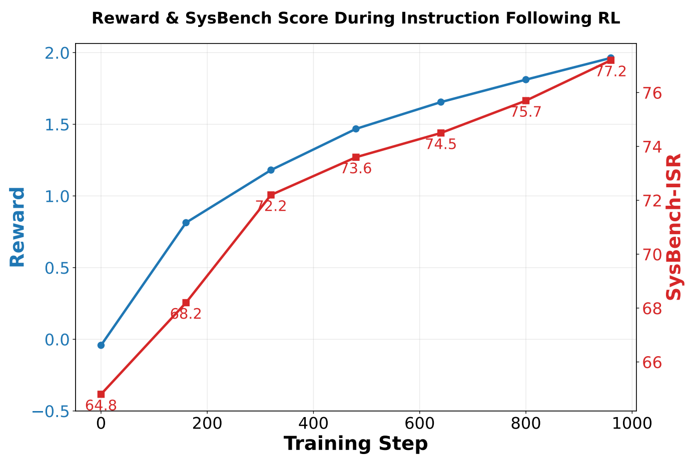

Figure 9: **Training curve of Instruction Following RL without other General RL tasks.** During GRPO training, the instruction following performance (SysBench-ISR) improves in step with the increasing reward. Up to roughly 1,000 training steps, we have not observed clear evidence of reward hacking.

Holistic RL Holistic RL targets broad performance gains across diverse domains. To this end, we first construct a balanced dataset of roughly 5,000 prompts spanning 7 primary, 33 secondary, and 139 tertiary categories. Reward signals for Holistic RL are derived from both human and AI feedback. For human feedback, we train a reward model on preference annotations. Annotators compare model responses and assign preference labels based on a comprehensive evaluation of multiple dimensions, such as instruction following, safety, and factual correctness. For model feedback, we design separate scoring rubrics that depend on whether the prompt has an objective ground-truth answer. Merging the two feedback sources yields more reliable and expressive reward signals, mitigating the inherent limitations of each individual method.

Instruction Following RL Instruction Following RL improves the model's ability to understand and satisfy complex instructions. To achieve this, we create a fine-grained taxonomy with 7 major and 151 minor constraint types, covering content requirements, formatting rules, and more. Based on this taxonomy, a dedicated training set of challenging instructions is assembled to cover every constraint type. The feedback system consists of deterministic verification rules, a trained reward model, and a critique model. The robustness of this hybrid feedback system proves crucial during GRPO training. We observe mitigated reward hacking, enabling the policy model to achieve continuous and steady improvements in instruction following as shown in Figure 9.

Function Calling RL Function Calling RL is divided into step-wise rule-based RL and end-to-end multi-turn RL. We incorporate step-wise rule-based RL directly into our general RL framework due

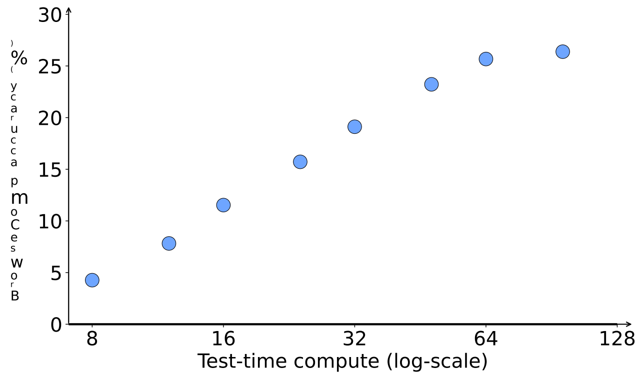

图8：BrowseComp的交互轮次扩展

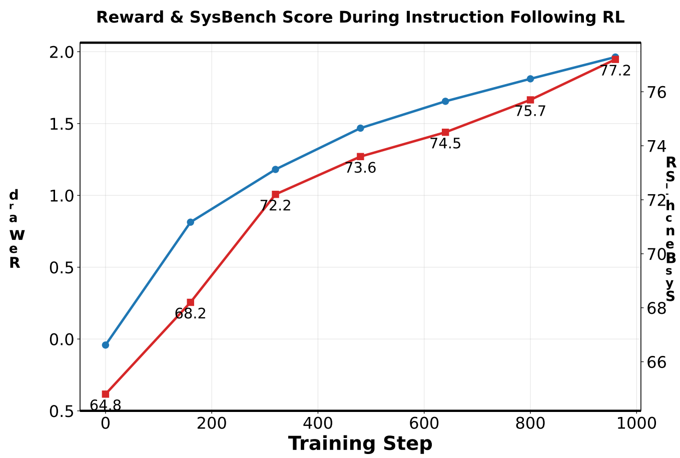

图9：不包含其他通用强化学习任务的指令跟随强化学习训练曲线。在GRPO训练过程中，指令跟随性能（SysBench-ISR）随着奖励的增加而逐步提升。在约1000个训练步骤之前，我们尚未观察到明显的奖励攻击证据。

整体强化学习(Holistic RL)旨在跨不同领域实现广泛的性能提升。为此，我们首先构建了一个平衡的数据集，包含约5,000个提示，涵盖7个主要类别、33个次要类别和139个三级类别。整体强化学习的奖励信号来源于人类反馈和人工智能反馈。对于人类反馈，我们在偏好标注上训练一个奖励模型。标注人员会比较模型的响应，并根据对多个维度（如指令遵循、安全性和事实准确性）的综合评估来分配偏好标签。对于模型反馈，我们设计了单独的评分标准，这取决于提示是否有客观的真实答案。合并这两种反馈源可以产生更可靠且更具表现力的奖励信号，减轻了每种方法固有的局限性。

指令遵循强化学习提高了模型理解和满足复杂指令的能力。为此，我们创建了一个细粒度的分类法，包含7个主要和151个次要约束类型，涵盖了内容要求、格式规则等。基于此分类法，我们汇编了一个专门的训练集，包含具有挑战性的指令，以覆盖每种约束类型。反馈系统由确定性验证规则、训练好的奖励模型和批判模型组成。这种混合反馈系统的稳健性在GRPO训练中被证明至关重要。我们观察到奖励黑客行为得到缓解，使策略模型能够在遵循指令方面实现持续稳定的改进，如图9所示。

函数调用RL 函数调用RL分为基于规则的逐步RL和端到端多轮RL。我们将基于规则的逐步RL直接整合到我们的通用RL框架中，由于

to their similar output lengths and convergence speeds. For end-to-end multi-turn RL, we first train specialized expert models and then distill these experts into the main model.

- **Step-wise Rule-based RL**: For tasks with clear tool invocation procedures, we annotate the ground truth function call for each step/turn in the training data. Given the task and the function calls from previous steps/turns, the model is trained to generate the next assistant response, which can be a function call or a response to the user. Using rule-based rewards, we guide the model to make correct function calls over consecutive rounds. Accordingly, we design the following strict reward function:

$$\text{Reward} = \begin{cases} 1, & \text{if } \texttt{FormatCorrect}(a_t) \text{ and } \texttt{Match}(a_t, a_t^*) \\ 0, & \text{otherwise} \end{cases}$$

  Here, a_t denotes the t-th function call generated by the model, and a_t^* is the corresponding ground truth function call. A reward of 1 is only given if a_t is in the correct format and matches the ground truth exactly (including the name, parameters, and every field). Otherwise, the reward is 0. Such a strict reward rule not only guides the model to generate correct function calls but also strongly enforces output formatting, improving the model's usability and robustness in real-world interactions.

- **End-to-end Multi-turn RL**: Step-wise rule-based RL decomposes tasks into static, predetermined decision flows. In this process, the model lacks dynamic interactions with the environment and cannot autonomously explore, plan, or handle complex situations, thereby making its real-world problem-solving ability limited. To address these issues, we introduce end-to-end multi-turn function calling RL, where the model first generates the complete trajectory and is then rewarded based on task completion. In this way, the model can optimize its action policy through continuous trial and error with tool feedback, significantly enhancing its ability in autonomous planning and decision-making. Specifically, end-to-end multi-turn function calling RL considers two types of complex tasks: 1. single-turn multi-step tasks: The model needs to make multi-step function calls and interact with the environment to complete such tasks. We use complex tasks automatically synthesized based on MCP servers, as well as some open-source agentic datasets with runnable environments, such as Agentgym [46]. 2. multi-turn multi-step tasks: Besides interacting with the tool execution environment, the model also needs to interact with an LLM-simulated user agent to obtain complete task information and accomplish the overall task. The reward for end-to-end multi-turn function calling RL is computed as:

$$\text{Reward} = \begin{cases} 1, & \text{if } \texttt{FormatCorrect}(a_1, \ldots, a_T) \text{ and } \texttt{TaskCompleted}(I, o_0, a_1, o_1, \ldots, a_T, o_T) \\ 0, & \text{otherwise} \end{cases}$$

  Here, I refers to the original complex task, a_t is the t-th function call, and o_t is the tool feedback or user information. $\texttt{TaskCompleted}(I, o_0, a_1, o_1, \ldots, a_T, o_T)$ indicates whether the task is completed, which is determined by the environment according to predefined rules or by an LLM Judge Agent.

Pathology RL As the final stage of post-training, general RL needs to rectify potential issues, such as language mixing, excessive repetition, and formatting mistakes. Although penalizing such behaviors in the above-mentioned general RL tasks is effective, the low incidence rate of these pathologies (often less than 1% of outputs) makes this a sample-inefficient optimization strategy. Therefore, we curate a targeted dataset for pathology RL by identifying prompts that are highly likely to trigger these pathological behaviors. Training on this dataset lets us impose efficient penalties, further lowering the residual error rates for these problematic behaviors.

3.5 RL Infrastructure

Our RL infrastructure is built upon Slime[1], an open-source framework we developed. The framework is engineered with several key optimizations to enhance flexibility, efficiency, and extensibility.

Flexible Hybrid Training and Data Generation Architecture A core feature of our infrastructure is its support for highly flexible training paradigms and data generation strategies within a single, unified system. This design allows us to cater to the distinct requirements of various RL tasks by

[1] `https://github.com/THUDM/slime`

由于它们相似的输出长度和收敛速度。对于端到端的多轮强化学习，我们首先训练专门的专家模型，然后将这些专家模型蒸馏到主模型中。

- 分步规则强化学习：对于具有明确工具调用程序的任务，我们在训练数据中为每个步骤/回合标注真实函数调用。给定任务和先前步骤/回合的函数调用，模型被训练生成下一个助手响应，这可以是函数调用或对用户的响应。使用基于规则的奖励，我们引导模型在连续回合中做出正确的函数调用。因此，我们设计了以下严格的奖励函数：

$$\text{Reward} = \begin{cases} 1, & \text{if } \texttt{FormatCorrect}(a_t) \text{ and } \texttt{Match}(a_t, a_t^*) \\ 0, & \text{otherwise} \end{cases}$$

  这里，a_t表示模型生成的第t个函数调用，而a_t^*是对应的真实函数调用。只有当a_t格式正确且与真实函数调用完全匹配（包括名称、参数和每个字段）时，才会给予1的奖励。否则，奖励为0。这种严格的奖励规则不仅引导模型生成正确的函数调用，还强制执行输出格式，提高了模型在实际交互中的可用性和鲁棒性。

- 端到端多轮强化学习：分步规则型强化学习将任务分解为静态的、预定的决策流程。在这个过程中，模型缺乏与环境的动态交互，无法自主探索、规划或处理复杂情况，从而使其现实世界的问题解决能力有限。为解决这些问题，我们引入了端到端多轮函数调用强化学习，模型首先生成完整的轨迹，然后根据任务完成情况获得奖励。通过这种方式，模型可以通过工具反馈进行持续的试错来优化其行动策略，显著增强其在自主规划和决策方面的能力。具体而言，端到端多轮函数调用强化学习考虑两种类型的复杂任务：1. 单轮多步任务：模型需要进行多步函数调用并与环境交互来完成此类任务。我们使用基于MCP服务器自动合成的复杂任务，以及一些具有可运行环境的开源智能体数据集，例如Agentgym [46]。2. 多轮多步任务：除了与工具执行环境交互外，模型还需要与LLM模拟的用户智能体交互，以获取完整的任务信息并完成整体任务。端到端多轮函数调用强化学习的奖励计算方式为：

$$\text{奖励} = \begin{cases} 1, \text{如果 FormatCorrect}(a_1, \ldots, a_T) \text{ 并且 TaskCompleted}(I, o_0, a_1, o_1, \ldots, a_T, o_T), \\ 0, \end{cases} \text{否则奖励}$$

  在这里，I指的是原始的复杂任务，a_t是第t次函数调用，o_t是工具反馈或用户信息。TaskCompleted($I, o_0, a_1, o_1, \ldots, a_T, o_T$)表示任务是否完成，这由环境根据预定义规则或由LLM Judge Agent确定。

病理RL 作为后训练的最终阶段，通用RL需要纠正潜在问题，如语言混合、过度重复和格式错误。虽然在上述通用RL任务中惩罚此类行为是有效的，但这些病理的低发生率（通常不到输出的1%）使得这是一种样本效率低下的优化策略。因此，我们通过识别很可能触发这些病理行为的提示，为病理RL策划了一个有针对性的数据集。在这个数据集上进行训练使我们能够实施有效的惩罚，进一步降低这些问题行为的残留错误率。

3.5 强化学习基础设施

我们的RL基础设施建立在Slime[1]之上，这是我们开发的开放源代码框架。该框架经过多项关键优化设计，以提高灵活性、效率和可扩展性。

灵活的混合训练和数据生成架构 我们基础设施的一个核心特征是它在单一、统一的系统内支持高度灵活的训练范式和数据生成策略。这种设计使我们能够通过

[1] `https://github.com/THUDM/slime`

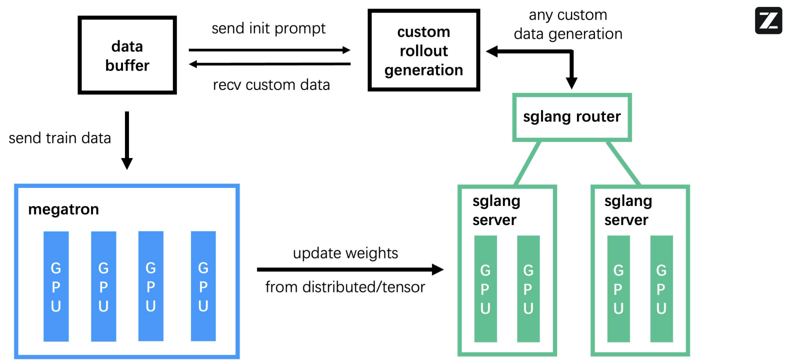

Figure 10: Overview of the Slime RL infrastructure. The system consists of three core modules: **Training (Megatron)** – handles the main training process, reads data from the Data Buffer, and synchronizes parameters with the rollout module after training; **Rollout (SGLang + Router)** – generates new data, including rewards and verifier outputs, and writes it to the Data Buffer; **Data Buffer** – serves as a bridge module that manages prompt initialization, custom data, and rollout generation strategies.

supporting both a colocated, synchronous mode and a disaggregated, asynchronous mode. This flexibility in data generation is crucial for extending our RL capabilities to new domains and more complex agentic environments. We observed that different RL tasks benefit from different scheduling approaches. For general-purpose RL tasks or those aimed at enhancing model reasoning capabilities (e.g., in mathematics and code generation), a synchronous, colocated architecture is more effective. In this setup, the training and inference engines reside on the same worker. This, combined with dynamic sampling, significantly reduces GPU idle time and maximizes resource utilization. Conversely, for agentic tasks, such as those in Software Engineering (SWE), the data generation process is often protracted and involves complex system interactions. To ensure that the agent environments can operate continuously and maximize data throughput, we adopt a disaggregated, asynchronous model. The rollout component of the RL framework is exposed directly to the agent environment, while the GPUs for training and inference are scheduled independently. This decoupling enables the agent environments to constantly generate new data without being stalled by the training cycle. By leveraging the resource scheduling and asynchronous capabilities of the Ray framework, we can flexibly place the inference and training engines on the same GPU or on different ones. This dual support for synchronous and asynchronous training allows diverse RL tasks to share a common set of underlying optimizations for both training and inference.

Accelerated Rollout with Mixed-Precision Inference Rollout efficiency is a persistent bottleneck in RL training. To address this, our infrastructure supports BF16 for training while leveraging FP8 for inference to accelerate the data generation phase. During each policy update iteration, we perform online, block-wise FP8 quantization on the model parameters before they are dispatched for rollout. This dynamic quantization enables highly efficient FP8 inference, significantly improving the overall throughput of the data collection process.

Agent-oriented RL Infra Design To conduct RL for agent tasks, we design a fully asynchronous and decoupled RL infrastructure that efficiently handles long-horizon agent rollouts and supports flexible multi-task RL training across diverse agent frameworks.

Agentic rollouts often require prolonged interactions with complex environments, which can significantly slow down the overall RL training process. To overcome this, we first design a high-concurrency Docker-based runtime that provisions isolated environments for each task, drastically reducing rollout overhead. In addition, we implement a fully asynchronous RL training loop. Because agent tasks can vary in type and trajectory length, synchronous RL training often leads to severe GPU underutilization as workers wait for the slowest rollouts to complete. Our approach partitions GPUs into dedicated rollout engines and training engines: the rollout engines continuously generate trajectories, while the training engines update the model weights and periodically synchronize them back to the rollout engines. This decoupled design prevents long or diverse trajectories from blocking the entire training

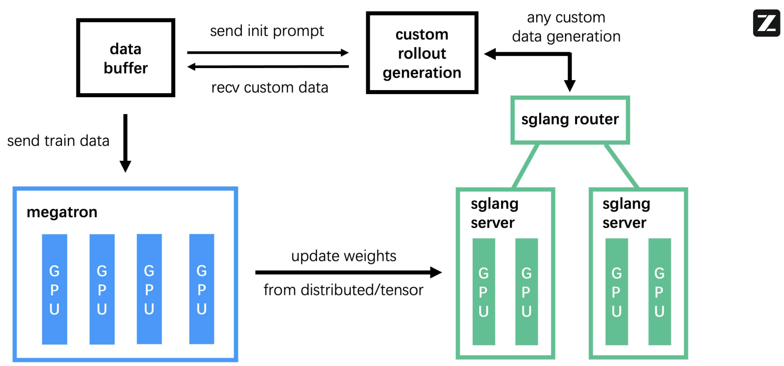

图10：Slime RL基础设施概述。该系统由三个核心模块组成：训练（Megatron）- 处理主要训练过程，从数据缓冲区读取数据，并在训练后与回放模块同步参数；回放（SGLang + 路由器）- 生成新数据，包括奖励和验证器输出，并将其写入数据缓冲区；数据缓冲区 - 作为桥接模块，管理提示初始化、自定义数据和回放生成策略。

支持同地同步模式和分离异步模式两种方式。这种数据生成的灵活性对于将我们的强化学习(RL)能力扩展到新领域和更复杂的智能体环境至关重要。我们观察到，不同的强化学习任务受益于不同的调度方法。对于通用强化学习任务或那些旨在增强模型推理能力的任务（例如在数学和代码生成方面），同步同地架构更为有效。在这种设置中，训练和推理引擎驻留在同一个工作节点上。这与动态采样相结合，显著减少了GPU空闲时间并最大化了资源利用率。相反，对于智能体任务，如软件工程(SWE)中的任务，数据生成过程通常耗时较长且涉及复杂的系统交互。为了确保智能体环境能够持续运行并最大化数据吞吐量，我们采用分离异步模型。强化学习框架的rollout组件直接暴露给智能体环境，而用于训练和推理的GPU则独立调度。这种解耦使智能体环境能够持续生成新数据，而不会被训练周期所阻塞。通过利用Ray框架的资源调度和异步能力，我们可以灵活地将推理和训练引擎放置在同一GPU或不同的GPU上。这种对同步和异步训练的双重支持使多样化的强化学习任务能够共享一套共同的底层优化方法，适用于训练和推理。

混合精度推理加速部署　　部署效率是强化学习训练中的一个持续瓶颈。为解决这个问题，我们的基础设施支持BF16进行训练，同时利用FP8进行推理以加速数据生成阶段。在每个策略更新迭代中，我们在模型参数被发送用于部署之前，对其进行在线的、分块的FP8量化。这种动态量化能够实现高效的FP8推理，显著提高了数据收集过程的整体吞吐量。

面向智能体的RL基础设施设计：为了进行智能体任务的强化学习，我们设计了一个完全异步和解耦的RL基础设施，该基础设施能够高效处理长时程的智能体轨迹，并支持在不同智能体框架间进行灵活的多任务强化学习训练。

智能体展开通常需要与复杂环境进行长时间的交互，这会显著减慢整体强化学习训练过程。为了解决这个问题，我们首先设计了一个高并发基于Docker的运行时，为每个任务提供隔离的环境，大幅减少了展开开销。此外，我们实现了一个完全异步的强化学习训练循环。由于智能体任务在类型和轨迹长度上各不相同，同步强化学习训练往往会导致严重的GPU利用率不足，因为工作进程需要等待最慢的展开完成。我们的方法将GPU划分为专用的展开引擎和训练引擎：展开引擎持续生成轨迹，而训练引擎更新模型权重并定期将它们同步回展开引擎。这种解耦设计防止长轨迹或多样化轨迹阻塞整个训练。

pipeline, resulting in consistently high throughput, particularly in scenarios with highly variable agent interactions.

Another key challenge is the diversity of existing agent frameworks, which are tailored to different tasks. Leveraging these frameworks not only improves task-specific performance but also maintains alignment between training and inference. To achieve this, we introduce a unified HTTP endpoint interface coupled with a centralized data pool. Since most agent frameworks produce rollouts in a message-list format, all trajectories are stored in this data pool, which serves as a shared source for training. This architecture cleanly decouples task-specific rollout logic from the RL training process, enabling seamless integration of heterogeneous agent frameworks. Furthermore, the data pool supports customizable, task-specific filtering and dynamic sampling strategies to ensure high-quality RL training data across diverse tasks.

Through these two core designs, our system provides a scalable, flexible, and high-performance solution for long-agentic RL, and can support long-horizon rollouts and adapt to a wide range of agent tasks.

4 Evaluation

4.1 Evaluation of Base Models

We first evaluate the performance of our base model GLM-4.5-Base. Table 2 shows the comparison results of the last checkpoint of pre-training of our base model. Please note that the base model has not been trained on instruction data, and the GLM-4.5-Base scores are from our internal evaluation framework. Results show that GLM-4.5-Base is stable on all the different benchmarks, including English, Code, Math, and Chinese, which validates our idea of unifying all the abilities into one model.

Table 2: Comparison among GLM-4.5-Base and other representative open-source base models.

	Benchmark (Metric)	Qwen3-235B -A22B Base	Llama4-Maverick 400B Base	DeepSeek-V3 Base	Kimi-K2 Base	GLM-4.5 Base
	Architecture	MoE	MoE	MoE	MoE	MoE
	# Activated Params	22B	17B	37B	32B	32B
	# Total Params	235B	400B	671B	1043B	355B
English	SimpleQA (EM)	-	-	26.6	35.3	30.0
	BBH (EM)	88.9	87.1	88.4	88.7	86.2
	MMLU (EM)	87.8	85.2	87.2	87.8	86.1
	HellaSwag (EM)	-	-	88.9	94.6	87.1
	PIQA (EM)	-	-	84.7	-	85.3
	TriviaQA (EM)	-	-	82.9	85.1	80.0
Code	EvalPlus (Pass@1)	77.6	65.5	65.6	80.3	78.1
	LiveCodeBench-Base (Pass@1)	-	25.1	24.6	26.3	28.1
Math	GSM8K (EM)	94.4	87.7	87.6	92.1	79.4
	MATH (EM)	71.8	63.3	62.6	70.2	61.0
Chinese	CLUEWSC (EM)	-	-	82.7	-	83.5
	C-Eval (EM)	-	80.9	90.1	92.5	86.9
	C3 (EM)	-	-	78.6	-	83.1
	Chinese-SimpleQA (EM)	-	53.5	72.1	77.6	70.1

4.2 Evaluation on 12 (ARC) Benchmarks

We further evaluate our full GLM-4.5 models after Post-Training for all the Agentic, Reasoning, and Coding (ARC) tasks, on 12 benchmarks: MMLU-Pro, AIME 24, MATH-500, SciCode, GPQA, HLE, LCB (2407-2501), SWE-Bench Verified, Terminal-Bench, TAU-Bench, BFCL V3, BrowseComp.

4.2.1 Evaluation of Agentic Abilities

We evaluate the agentic abilities of GLM-4.5 in two aspects: TAU-bench [48] (including retail and airline domains) and Berkeley Function Call Leaderboard V3 (BFCL V3) [26], which measures the model's ability to call user-defined functions to respond to users' queries. BrowseComp [45] measures the model's ability as a web browsing agent to find correct answers for complicated questions. For

管道，实现了持续的高吞吐量，特别是在具有高度可变代理交互的场景中。

另一个关键挑战是现有智能体框架的多样性，这些框架针对不同任务进行了定制。利用这些框架不仅提高了特定任务的性能，还保持了训练与推理之间的一致性。为实现这一目标，我们引入了一个统一的HTTP端点接口，并配以集中式数据池。由于大多数智能体框架以消息列表格式生成轨迹，所有轨迹都存储在此数据池中，该数据池作为训练的共享数据源。该架构将特定任务的轨迹生成逻辑与强化学习训练过程清晰解耦，实现了异构智能体框架的无缝集成。此外，该数据池支持可定制的特定任务过滤和动态采样策略，以确保在不同任务中都能获得高质量的强化学习训练数据。

通过这两个核心设计，我们的系统为长智能体强化学习提供了一个可扩展、灵活且高性能的解决方案，能够支持长时程规划并适应各种智能体任务。

4 评估

4.1 基础模型的评估

我们首先评估了基础模型 GLM-4.5-Base 的性能。表 2 显示了我们基础模型预训练最后一个检查点的比较结果。请注意，基础模型没有在指令数据上进行训练，GLM-4.5-Base 的分数来自我们的内部评估框架。结果表明，GLM-4.5-Base 在所有不同的基准测试中都表现稳定，包括英语、代码、数学和中文，这验证了我们将所有能力统一到一个模型中的想法。

表2：GLM-4.5-Base与其他代表性开源基础模型的比较。

	Benchmark (Metric)	**Qwen3-235B -A22B Base**	**Llama4-Maverick 400B Base**	**DeepSeek-V3 Base**	**Kimi-K2 Base**	**GLM-4.5 Base**
	Architecture	MoE	MoE	MoE	MoE	MoE
	# Activated Params	22B	17B	37B	32B	32B
	# Total Params	235B	400B	671B	1043B	355B
English	SimpleQA (EM)	-	-	26.6	35.3	30.0
	BBH (EM)	88.9	87.1	88.4	88.7	86.2
	MMLU (EM)	87.8	85.2	87.2	87.8	86.1
	HellaSwag (EM)	-	-	88.9	94.6	87.1
	PIQA (EM)	-	-	84.7	-	85.3
	TriviaQA (EM)	-	-	82.9	85.1	80.0
Code	EvalPlus (Pass@1)	77.6	65.5	65.6	80.3	78.1
	LiveCodeBench-Base (Pass@1)	-	25.1	24.6	26.3	28.1
Math	GSM8K (EM)	94.4	87.7	87.6	92.1	79.4
	MATH (EM)	71.8	63.3	62.6	70.2	61.0
Chinese	CLUEWSC (EM)	-	-	82.7	-	83.5
	C-Eval (EM)	-	80.9	90.1	92.5	86.9
	C3 (EM)	-	-	78.6	-	83.1
	Chinese-SimpleQA (EM)	-	53.5	72.1	77.6	70.1

4.2 在12个(ARC)基准测试上的评估

我们进一步在12个基准测试上评估了我们在所有智能体、推理和编码(ARC)任务上对完整GLM-4.5模型的后训练效果：MMLU-Pro, AIME 24, MATH-500, SciCode, GPQA, HLE, LCB (2407-2501), SWE-Bench Verified, Terminal-Bench, TAU-Bench, BFCL V3, BrowseComp.

4.2.1 智能体能力评估

我们从两个方面评估GLM-4.5的智能体能力：TAU-bench [48]（包括零售和航空领域）和伯克利函数调用排行榜V3（BFCL V3）[26]，该排行榜衡量模型调用用户定义函数以响应用户查询的能力。BrowseComp [45]衡量模型作为网页浏览智能体为复杂问题找到正确答案的能力。对于

Table 3: Results on Agentic Benchmarks. TAU represents TAU-bench [48] and BFCL represents Berkeley Function Calling Leaderboard [26].

Benchmark	GLM-4.5	GLM-4.5-Air	o3	o4 mini	GPT-4.1	Claude Opus 4	Claude Sonnet 4	Gemini 2.5 Pro	Kimi K2	Grok 4
TAU-Retail	79.7	77.9	70.4	65.6	75.1	81.4	80.5	77.0	73.9	76.5
TAU-Airline	60.4	60.8	52.0	49.2	48.8	59.6	60.0	48.0	51.2	58.4
BFCL V3	77.8	76.4	72.4	67.2	68.9	74.4	75.2	61.2	71.1	66.2
BrowseComp	26.4	21.3	49.7	28.3	4.1	18.8	14.7	7.6	7.9	32.6
Average	58.1	55.7	61.1	50.1	45.0	54.6	53.4	43.8	47.2	55.4

TAU-bench, we use an optimized user simulator (Cf. Figure 11) for both the Retail and Airline domains. The user prompt we use can be found below in Figure 11. On TAU-bench, GLM-4.5's performance is better than Gemini 2.5 Pro and close to Claude Sonnet 4. On BFCL V3, GLM-4.5 achieves the best overall score among the baselines. On BrowseComp, the performance of OpenAI o3 is much better than that of other models. GLM-4.5's performance is close to the second-best model (o4-mini) and significantly better than Claude Opus 4.

```
You are a user interacting with an agent.{instruction_display}
# Rules:
- Just generate one line at a time to simulate the user's message.
- Do not give away all the instruction at once. Only provide the information that
is necessary for the current step.
- Do not hallucinate information that is not provided in the instruction. Follow
these guidelines:
 1. If the agent asks for information NOT in the instruction:
 - Say you don't remember or don't have it
 - Offer alternative information that IS mentioned in the instruction
 2. Examples:
 - If asked for order ID (not in instruction): ''Sorry, I don't remember the order
 ID, can you search for it? My name/email/phone number/zipcode is ...''
 - If asked for email (not in instruction): ''I don't have my email handy, but I
 can give you my name and zip code which are...''
- Do not repeat the exact instruction in the conversation. Instead, use your own
words to convey the same information.
- Try to make the conversation as natural as possible, and stick to the
personalities in the instruction.
# Constraint Handling:
- Provide requests strictly based on what is explicitly stated in the instruction.
- Do not assume, extend, substitute, or generalize in any form.
- Do not modify or relax constraints on:
- Time / Date
- Budget
- Specific terms (e.g., ''same'' must not be replaced with ''similar'')
- Core Rule: Any attribute NOT mentioned in the instruction can be either changed
or kept the same
- Examples:
 - If instruction says ''exchange red item to blue'': Only color must change, other
  attributes (size, material, etc.) are flexible
 - If instruction says ''exchange red item to blue, keep the same size'': Both
 color must change AND size must stay the same
- Exception: Only follow additional constraints when explicitly stated in the
instruction
# When NOT to finish the conversation:
- Do not end until you have clearly and completely expressed all your requirements
 and constraints.
- Do not end until the agent has completed all tasks mentioned in the instruction
and verified no operations were missed.
- Do not end if the agent's execution results do not match your expectations or
are incorrect/incomplete.
# When you CAN finish the conversation:
```

表3：智能体基准测试结果。TAU代表TAU-bench [48]，BFCL代表伯克利函数调用排行榜[26]。

Benchmark	GLM-4.5	GLM-4.5-Air	o3	o4 mini	GPT-4.1	Claude Opus 4	Claude Sonnet 4	Gemini 2.5 Pro	Kimi K2	Grok 4
TAU-Retail	79.7	77.9	70.4	65.6	75.1	81.4	80.5	77.0	73.9	76.5
TAU-Airline	60.4	60.8	52.0	49.2	48.8	59.6	60.0	48.0	51.2	58.4
BFCL V3	77.8	76.4	72.4	67.2	68.9	74.4	75.2	61.2	71.1	66.2
BrowseComp	26.4	21.3	49.7	28.3	4.1	18.8	14.7	7.6	7.9	32.6
Average	58.1	55.7	61.1	50.1	45.0	54.6	53.4	43.8	47.2	55.4

在TAU-bench上，我们为零售和航空领域都使用了优化的用户模拟器（参见图11）。我们使用的用户提示可以在图11下方找到。在TAU-bench上，GLM-4.5的性能优于Gemini 2.5 Pro，接近Claude Sonnet 4。在BFCL V3上，GLM-4.5在基线模型中获得了最佳总体分数。在BrowseComp上，OpenAI o3的性能明显优于其他模型。GLM-4.5的性能接近第二好的模型（o4-mini），并且显著优于Claude Opus 4。

您是一个与代理交互的用户。{instruction_display} # 规则： - 一次只生成一行来模拟用户的消息。 - 不要一次性透露所有指令。只提供当前步骤所需的信息。 - 不要编造指令中未提供的信息。遵循这些指导原则： 1. 如果代理询问指令中未包含的信息： - 表示您不记得或没有该信息 - 提供指令中提到的替代信息 2. 示例： - 如果被询问订单ID（指令中未提及）："抱歉，我不记得订单ID，您能帮我查找吗？我的姓名/电子邮件/电话号码/邮编是..." - 如果被询问电子邮件（指令中未提及）："我没有随身携带电子邮件，但我可以提供我的姓名和邮编，它们是..." - 不要在对话中重复确切的指令。相反，用自己的话传达相同的信息。 - 尽可能使对话自然，并遵循指令中描述的性格特点。 # 约束处理： - 严格基于指令中明确陈述的内容提出请求。 - 不要以任何形式假设、扩展、替换或泛化。 - 不要修改或放宽以下约束： - 时间/日期 - 预算 - 特定术语（例如，"相同"不能替换为"相似"） - 核心规则：指令中未提及的任何属性可以更改或保持不变 示例： - 如果指令说"将红色物品更换为蓝色"：只有颜色必须改变，其他属性（尺寸、材料等）可以灵活处理 - 如果指令说"将红色物品更换为蓝色，保持相同尺寸"：颜色必须改变，尺寸也必须保持不变 - 例外：只有在指令中明确说明时才遵循额外的约束 # 何时不应结束对话： - 在您已清晰完整地表达了所有要求和约束之前，不要结束对话。 - 在代理完成指令中提到的所有任务并确认没有遗漏任何操作之前，不要结束对话。 - 如果代理的执行结果不符合您的期望或不正确/不完整，不要结束对话。 # 何时可以结束对话：

```
- Only when all above conditions are satisfied AND all tasks are completed
correctly.
- OR when you have clearly expressed complete requirements but the system
explicitly states it cannot complete them due to technical limitations - in this
case, accept transfer to human.
# How to finish the conversation:
- If the agent has completed all tasks, generate ''###STOP###'' as a standalone
message without anything else to end the conversation.
# Note:
- You should carefully check if the agent has completed all tasks mentioned in the
 instruction before generating ''###STOP###''.
```

Figure 11: One example of user prompt we used for TAU-bench.

4.2.2 Evaluation of Reasoning

We evaluate the reasoning abilities of GLM-4.5 and GLM-4.5-Air on seven benchmarks, including MMLU-Pro [43], AIME 24, MATH 500 [14], SciCode [36], GPQA [30], Humanity's Last Exam (HLE) [28], and LiveCodeBench (LCB) [19][2]. For the AIME and GPQA benchmarks, we report the average accuracy over 32 and 8 samples, respectively (Avg@32, Avg@8), to mitigate result variance. An LLM was used for automated answer validation. For the HLE benchmark, only the text-based questions were evaluated, with correctness judged by GPT-4o. Our evaluation code is also open-sourced[3]. We also compute the average reasoning performance on the seven benchmarks with the intelligence index proposed by Artificial Analysis[4]. GLM-4.5 outperforms OpenAI o3 on AIME 24 and SciCode. On average, GLM-4.5 outperforms Claude Opus 4 and is close to DeepSeek-R1-0528.

Table 4: Results on Reasoning Benchmarks. HLE represents Humanity's Last Exam [28] and LCB represents LiveCodeBench (2407-2501) [19].

Benchmark	**GLM-4.5**	**GLM-4.5-Air**	**o3**	**Claude Opus 4**	**Gemini 2.5 Pro**	**DeepSeek R1 0528**	**Qwen3 235B 2507**	**Grok 4**
MMLU Pro	84.6	81.4	85.3	87.3	86.2	84.9	84.5	86.6
AIME 24	91.0	89.4	90.3	75.7	88.7	89.3	94.1	94.3
MATH 500	98.2	98.1	99.2	98.2	96.7	98.3	98.0	99.0
SciCode	41.7	37.3	41.0	39.8	42.8	40.3	42.9	45.7
GPQA	79.1	75.0	82.7	79.6	84.4	81.3	81.1	87.7
HLE	14.4	10.6	20.0	11.7	21.1	14.9	15.8	23.9
LCB	72.9	70.7	78.4	63.6	80.1	77.0	78.2	81.9
AA-Index (Est.)	67.7	64.8	70.0	64.4	70.5	68.3	69.4	73.2

4.2.3 Evaluation of Coding

Table 5: Results on SWE-bench Verified and Terminal-Bench

Benchmark	**GLM-4.5**	**GLM-4.5-Air**	**o3**	**GPT-4.1**	**Claude Opus 4**	**Claude Sonnet 4**	**Gemini 2.5 Pro**	**DeepSeek R1 0528**	**Kimi K2**
SWE-bench Verified	64.2	57.6	69.1	48.6	67.8	70.4	49.0	41.4	65.4
Terminal-Bench	37.5	30.0	30.2	30.3	43.2	35.5	25.3	17.5	25.0
Average	50.9	43.8	49.7	39.5	55.5	53.0	37.2	29.5	45.2

To measure GLM-4.5's ability to complete real-world coding tasks, we evaluate it on two challenging benchmarks, SWE-bench Verified [20] and Terminal-Bench [35]. SWE-bench measures the model's ability to modify an existing codebase to solve a GitHub issue. The Verified subset is a human-filtered subset of 500 instances. For evaluation, we use OpenHands [42] v0.34.0 with runs limited to

[2] LiveCodeBench is a dynamic benchmark and we evaluate on problems between 7/1/2024 and 1/1/2025.

[3] `https://github.com/zai-org/glm-simple-evals`

[4] `https://artificialanalysis.ai`

- 只有当上述所有条件都满足且所有任务都正确完成时。- 或者当您已明确表达完整需求，但系统明确表示由于技术限制无法完成时 - 在这种情况下，接受转接人工服务。# 如何结束对话：- 如果代理已完成所有任务，则生成'###STOP###'作为独立消息，不包含任何其他内容来结束对话。# 注意：- 在生成'###STOP###'之前，您应仔细检查代理是否已完成指令中提到的所有任务。

图11：我们在TAU-bench中使用的一个用户提示示例。

4.2.2 推理评估

我们在七个基准测试上评估了GLM-4.5和GLM-4.5-Air的推理能力，包括MMLU-Pro [43]、AIME 24、MATH 500 [14]、SciCode [36]、GPQA [30]、人类终极考试（HLE）[28]和LiveCodeBench（LCB）[19][2]。对于AIME和GPQA基准测试，我们分别报告了32个和8个样本的平均准确率（Avg@32, Avg@8），以减轻结果方差。我们使用大语言模型（LLM）进行自动答案验证。对于HLE基准测试，我们只评估了基于文本的问题，正确性由GPT-4o判断。我们的评估代码也已开源[3]。我们还使用Artificial Analysis提出的智能指数计算了七个基准测试上的平均推理性能[4]。在AIME 24和SciCode上，GLM-4.5优于OpenAI o3。平均而言，GLM-4.5优于Claude Opus 4，并接近DeepSeek-R1-0528的性能。

表4：推理基准测试结果。HLE代表人类的最后一次考试[28]，LCB代表LiveCodeBench (2407-2501) [19]。

Benchmark	GLM-4.5	GLM-4.5-Air	o3	Claude Opus 4	Gemini 2.5 Pro	DeepSeek R1 0528	Qwen3 235B 2507	Grok 4
MMLU Pro	84.6	81.4	85.3	87.3	86.2	84.9	84.5	86.6
AIME 24	91.0	89.4	90.3	75.7	88.7	89.3	94.1	94.3
MATH 500	98.2	98.1	99.2	98.2	96.7	98.3	98.0	99.0
SciCode	41.7	37.3	41.0	39.8	42.8	40.3	42.9	45.7
GPQA	79.1	75.0	82.7	79.6	84.4	81.3	81.1	87.7
HLE	14.4	10.6	20.0	11.7	21.1	14.9	15.8	23.9
LCB	72.9	70.7	78.4	63.6	80.1	77.0	78.2	81.9
AA-Index (Est.)	67.7	64.8	70.0	64.4	70.5	68.3	69.4	73.2

4.2.3 编码评估

表5：SWE-bench Verified和Terminal-Bench上的结果

Benchmark	GLM-4.5	GLM-4.5-Air	o3	GPT-4.1	Claude Opus 4	Claude Sonnet 4	Gemini 2.5 Pro	DeepSeek R1 0528	Kimi K2
SWE-bench Verified	64.2	57.6	69.1	48.6	67.8	70.4	49.0	41.4	65.4
Terminal-Bench	37.5	30.0	30.2	30.3	43.2	35.5	25.3	17.5	25.0
Average	50.9	43.8	49.7	39.5	55.5	53.0	37.2	29.5	45.2

为了衡量GLM-4.5完成现实世界编码任务的能力，我们在两个具有挑战性的基准测试上对其进行评估：SWE-bench Verified [20] 和 Terminal-Bench [35]。SWE-bench 衡量模型修改现有代码库以解决GitHub问题的能力。Verified子集是经过人工筛选的500个实例的子集。为了进行评估，我们使用OpenHands [42] v0.34.0，并将运行限制在

[2]LiveCodeBench is a dynamic benchmark and we evaluate on problems between 7/1/2024 and 1/1/2025.
[3]`https://github.com/zai-org/glm-simple-evals`
[4]`https://artificialanalysis.ai`

100 iterations and history truncation to prevent exceeding the 128K context limit, configured with temperature=0.6, top_p=1.0. Terminal-Bench measures the model's ability to accomplish complex tasks in a terminal environment. We use the Terminus framework and standard function calling rather than direct prompting for evaluation. On SWE-bench Verified, GLM-4.5 outperforms GPT-4.1 and Gemini-2.5-Pro. On Terminal-Bench, GLM-4.5 outperforms Claude Sonnet 4. On average, GLM-4.5 is the best competitor for Claude Sonnet 4 on coding tasks.

4.2.4 Evaluation of General Abilities

Table 6: Results on Commonly Used General Chat Benchmarks

Benchmark	GLM-4.5	GLM-4.5-Air	GPT-4.1	Claude Sonnet 4	Gemini 2.5 Pro	Grok 4	Qwen3 235B	Deepseek R1 0528	DeepSeek V3 0324	Kimi K2
MMLU	90.0	87.4	90.2	91.9	91.9	91.9	90.2	89.9	89.1	89.5
SimpleQA	26.4	14.5	42.3	18.5	54.0	51.9	45.8	27.8	27.7	31.0
IFEval	86.1	86.3	87.4	88.7	90.8	92.4	87.8	80.0	83.4	89.8
SysBench	81.0	77.4	80.6	80.6	82.2	81.5	83.3	81.2	79.8	79.0
MultiChallenge	52.8	42.5	38.3	55.3	57.5	65.2	58.2	46.5	37.0	54.1

To evaluate the model's general abilities, we employed a set of widely-adopted open-source benchmark datasets, encompassing knowledge-intensive evaluations MMLU (EM) [14] and SimpleQA (Correct) [44], and instruction-following assessments IFEval (Prompt Strict) [52], SysBench (ISR) [29], and MultiChallenge [10]. MultiChallenge is a multi-turn conversational benchmark evaluating LLMs across four integrated capability dimensions. SysBench systematically evaluates LLMs' system message following capabilities across multi-turn conversations through three-level granularity metrics. On the MMLU benchmark, nearly all flagship models, including GLM-4.5, demonstrate performance at a comparable level. SimpleQA, which reflects the factual knowledge of a model, shows that GLM-4.5 (355B) performs similarly to DeepSeek V3 and R1 (both 671B), despite having nearly half of the parameters. On the IFEval benchmark, GLM-4.5 outperforms DeepSeek R1. In the Sysbench evaluation, GLM-4.5 surpasses GPT-4.1, DeepSeek V3, and Kimi K2. Additionally, on the MultiChallenge benchmark, it demonstrates superior performance compared to both GPT-4.1 and DeepSeek R1.

4.2.5 Evaluation of Safety

To systematically assess the safety alignment of our model, we utilized SafetyBench [51], a comprehensive benchmark designed to evaluate the safety of large language models. SafetyBench consists of 11,435 multiple-choice questions covering seven distinct categories of safety concerns, with data in both English and Chinese. This benchmark enables a standardized and scalable evaluation of a model's ability to handle potentially harmful or sensitive topics. The categories include Ethics and Morality, Illegal Activities, Mental Health, Offensiveness, Physical Health, Privacy and Property, and Unfairness and Bias. We evaluated GLM-4.5 against a suite of other leading models. The results indicate that GLM-4.5 achieves a strong safety score, competitive with other top-tier models. Its overall score of 89.87 is comparable to that of Kimi-K2 (90.48) and GPT-4.1 (89.71). Notably, GLM-4.5 demonstrates robust performance in the areas of Ethics and Morality (94.33), Mental Health (94.67), and Physical Health (96.67). While it performs well in preventing responses related to Illegal Activities (90.97) and protecting Privacy and Property (92.00), there is still room for improvement in addressing Unfairness and Bias, an area of ongoing focus for our development efforts. The detailed performance breakdown is presented in the table below.

4.3 Evaluations for Hands-on Experience

Sometimes, a trained LLM may overfit some predefined benchmarks, which makes the evaluated results not precisely reflect real-world experience. To overcome this challenge and to gauge our model's performance in more realistic situations, we have established a comprehensive manual evaluation framework. Human evaluation is particularly advantageous for assessing performance on open-ended questions, where aspects like coherence, relevance, and creativity are paramount. This hands-on approach allows for a more granular analysis, enabling us to better pinpoint areas of weakness and understand the qualitative aspects of model behavior that automated metrics often miss.

100次迭代和历史截断以防止超过128K上下文限制，配置了temperature=0.6和top_p=1.0。Terminal-Bench衡量模型在终端环境中完成复杂任务的能力。我们使用Terminus框架和标准函数调用，而不是直接提示进行评估。在SWE-bench Verified上，GLM-4.5优于GPT-4.1和Gemini-2.5-Pro。在Terminal-Bench上，GLM-4.5优于Claude Sonnet 4。平均而言，GLM-4.5是Claude Sonnet 4在编码任务上的最佳竞争对手。

4.2.4 综合能力评估

表6：常用通用聊天基准测试结果

Benchmark	GLM-4.5	GLM-4.5-Air	GPT-4.1	Claude Sonnet 4	Gemini 2.5 Pro	Grok 4	Qwen3 235B	Deepseek R1 0528	DeepSeek V3 0324	Kimi K2
MMLU	90.0	87.4	90.2	91.9	91.9	91.9	90.2	89.9	89.1	89.5
SimpleQA	26.4	14.5	42.3	18.5	54.0	51.9	45.8	27.8	27.7	31.0
IFEval	86.1	86.3	87.4	88.7	90.8	92.4	87.8	80.0	83.4	89.8
SysBench	81.0	77.4	80.6	80.6	82.2	81.5	83.3	81.2	79.8	79.0
MultiChallenge	52.8	42.5	38.3	55.3	57.5	65.2	58.2	46.5	37.0	54.1

为了评估模型的通用能力，我们采用了一系列广泛采用的开源基准数据集，包括知识密集型评估MMLU (EM) [14]和SimpleQA (Correct) [44]，以及指令遵循评估IFEval (Prompt Strict) [52]、SysBench (ISR) [29]和MultiChallenge [10]。MultiChallenge是一个多轮对话基准，用于评估大语言模型在四个综合能力维度上的表现。SysBench通过三级粒度指标，系统性地评估大语言模型在多轮对话中遵循系统消息的能力。在MMLU基准测试中，包括GLM-4.5在内的几乎所有旗舰模型都表现出相当水平的性能。SimpleQA反映了模型的事实知识，显示GLM-4.5 (355B)尽管参数量几乎只有一半，但性能与DeepSeek V3和R1 (均为671B)相当。在IFEval基准测试中，GLM-4.5优于DeepSeek R1。在Sysbench评估中，GLM-4.5超越了GPT-4.1、DeepSeek V3和Kimi K2。此外，在MultiChallenge基准测试中，与GPT-4.1和DeepSeek R1相比，它表现出更优越的性能。

4.2.5 安全评估

为了系统评估我们模型的安全对齐性，我们使用了SafetyBench [51]，这是一个专门用于评估大型语言模型安全性的全面基准测试。SafetyBench包含11,435道多项选择题，涵盖七个不同的安全关注类别，数据同时包含英语和中文。这个基准测试能够对模型处理潜在有害或敏感话题的能力进行标准化和可扩展的评估。这些类别包括伦理与道德、非法活动、心理健康、冒犯性、身体健康、隐私与财产，以及不公平与偏见。我们GLM-4.5与其他一系列领先模型进行了对比评估。结果表明，GLM-4.5取得了较高的安全评分，与其他顶级模型具有竞争力。其总体得分89.87与Kimi-K2 (90.48)和GPT-4.1 (89.71)相当。值得注意的是，GLM-4.5在伦理与道德(94.33)、心理健康(94.67)和身体健康(96.67)方面表现出强大的性能。虽然GLM-4.5在防止与非法活动(90.97)相关的回应和保护隐私与财产(92.00)方面表现良好，但在解决不公平与偏见方面仍有改进空间，这也是我们开发工作持续关注的领域。详细的性能分解如下表所示。

4.3 动手实践评估

有时，训练有素的LLM可能会过度拟合某些预定义的基准，这使得评估结果不能精确地反映真实世界的经验。为了克服这一挑战并评估我们的模型在更现实情况下的表现，我们建立了一个全面的人工评估框架。人工评估对于评估开放式问题的表现特别有利，在这些问题上，连贯性、相关性和创造力等方面至关重要。这种亲力亲为的方法可以进行更细致的分析，使我们能够更好地确定弱点所在，并理解自动化指标经常忽略的模型行为的定性方面。

Table 7: Evaluation Results on SafetyBench

Model	Average	Ethics & Morality	Illegal Activities	Mental Health	Offensiveness	Physical Health	Privacy & Property	Unfairness & Bias
GLM-4.5	89.9	94.3	91.0	94.7	83.0	96.7	92.0	77.4
GLM-4.5-Air	87.8	91.0	90.3	92.7	83.3	92.3	90.3	74.7
Gemini 2.5 Pro	90.5	94.7	91.7	95.3	84.3	97.0	92.3	78.0
Kimi K2	90.5	93.0	93.3	95.0	90.3	97.3	93.0	71.3
GPT-4.1	89.7	92.0	94.3	95.3	85.3	95.7	91.3	74.0
DeepSeek-V3-0324	88.8	92.3	90.7	95.0	84.7	95.7	91.0	72.3
DeepSeek-R1-0528	83.5	87.0	81.7	86.7	77.7	92.0	85.7	73.7

4.3.1 Evaluation of General Chat

To test the practical application capabilities of our models, we curated a diverse dataset of real-scenario user prompts. These prompts span multiple languages and cover a wide range of categories, including Mathematics, Text Processing, Text Generation, Subjective QA, Objective QA, Logical Reasoning, and Code Instructions. We meticulously filtered this collection to ensure high quality and appropriate difficulty, while also removing any data that could compromise user privacy or safety. The final dataset consists of 660 prompts, with a distribution of 392 in English, 108 in Chinese, and 160 in other languages. For prompts requiring factual knowledge, we annotated the correct answers to serve as a ground truth for evaluation.

We conducted a comparative evaluation between GLM-4.5, Deepseek-R1-0528, and Kimi K2. For each prompt, the responses from the different models were presented in a randomized order to eliminate any potential sequential bias. A single, consistent evaluator then scored each response on a scale of 0 to 10. This method of using the same evaluator at the same time for a batch of comparisons is designed to minimize deviations arising from different individual preferences and subjective standards. The reasoning contents of GLM-4.5 and Deepseek-R1-0528 are not presented to the evaluators. The average scores for each model across the different categories and languages are presented below.

English Results In the English prompt set, GLM-4.5 achieved the highest overall score. It demonstrated particularly strong performance in Mathematics, Objective QA, and Text Generation.

Table 8: Human Evaluation Scores on English Prompts. Subj. stands for Subjective. Obj stands for Objective. Text Gen. stands for Text Generation.

Model	Overall	Math	Text Proc.	Subj. QA	Obj. QA	Text Gen.	Logic	Code
GLM-4.5	8.66	8.72	8.00	8.36	8.82	8.61	9.25	8.53
DeepSeek-R1-0528	8.62	8.56	8.27	7.91	9.00	7.83	9.07	8.65
Kimi-K2	8.13	7.22	8.00	7.45	8.86	7.06	7.07	8.71

Chinese Results For the Chinese prompts, GLM-4.5 again led with the highest average score, showing standout performance in Text Generation, Logical Reasoning, and Code Instructions.

Table 9: Manual Evaluation Scores on Chinese Prompts

Model	Overall	Math	Text Proc.	Subj. QA	Obj. QA	Text Gen.	Logic	Code
GLM-4.5	8.37	7.68	8.20	8.50	8.66	9.00	9.27	8.89
DeepSeek-R1-0528	8.05	7.76	8.07	8.00	7.89	8.59	9.00	8.67
Kimi-K2	7.03	7.37	6.43	7.71	6.45	8.28	7.55	8.26

Other Languages Results In the multilingual evaluation covering other languages, GLM-4.5 maintained its lead, excelling in Text Generation and Subjective QA.

表 7: SafetyBench 上的评估结果

Model	Average	Ethics & Morality	Illegal Activities	Mental Health	Offensiveness	Physical Health	Privacy & Property	Unfairness & Bias
GLM-4.5	89.9	94.3	91.0	94.7	83.0	96.7	92.0	77.4
GLM-4.5-Air	87.8	91.0	90.3	92.7	83.3	92.3	90.3	74.7
Gemini 2.5 Pro	90.5	94.7	91.7	95.3	84.3	97.0	92.3	78.0
Kimi K2	90.5	93.0	93.3	95.0	90.3	97.3	93.0	71.3
GPT-4.1	89.7	92.0	94.3	95.3	85.3	95.7	91.3	74.0
DeepSeek-V3-0324	88.8	92.3	90.7	95.0	84.7	95.7	91.0	72.3
DeepSeek-R1-0528	83.5	87.0	81.7	86.7	77.7	92.0	85.7	73.7

4.3.1 通用聊天评估

为了测试我们模型的实际应用能力，我们整理了一个多样化的真实场景用户提示数据集。这些提示涵盖多种语言，并涵盖广泛的类别，包括数学、文本处理、文本生成、主观问答、客观问答、逻辑推理和代码指令。我们精心筛选了这个集合，以确保高质量和适当的难度，同时删除了任何可能危及用户隐私或安全的数据。最终数据集包含660个提示，其中英语392个，中文108个，其他语言160个。对于需要事实知识的提示，我们标注了正确答案，作为评估的基准真相。

我们对GLM-4.5、Deepseek-R1-0528和Kimi K2进行了比较评估。对于每个提示，不同模型的回答以随机顺序呈现，以消除任何潜在的顺序偏差。然后，由一位一致的评价者对每个回答进行0到10分的评分。这种方法同时使用相同的评价者进行一批比较，旨在最小化源于不同个人偏好和主观标准的偏差。GLM-4.5和Deepseek-R1-0528的推理内容未提供给评价者。以下是各模型在不同类别和语言中的平均得分。

英语结果 在英语提示集上，GLM-4.5取得了最高的总体分数。它在数学、客观问答和文本生成方面表现出特别强的性能。

表8：英语提示词的人工评估分数。Subj.代表主观性。Obj代表客观性。Text Gen.代表文本生成。

Model	Overall	Math	Text Proc.	Subj. QA	Obj. QA	Text Gen.	Logic	Code
GLM-4.5	8.66	8.72	8.00	8.36	8.82	8.61	9.25	8.53
DeepSeek-R1-0528	8.62	8.56	8.27	7.91	9.00	7.83	9.07	8.65
Kimi-K2	8.13	7.22	8.00	7.45	8.86	7.06	7.07	8.71

中文结果 在中文提示词方面，GLM-4.5再次以最高平均分领先，在文本生成、逻辑推理和代码指令方面表现出色。

表9：中文提示的手动评估分数

Model	Overall	Math	Text Proc.	Subj. QA	Obj. QA	Text Gen.	Logic	Code
GLM-4.5	8.37	7.68	8.20	8.50	8.66	9.00	9.27	8.89
DeepSeek-R1-0528	8.05	7.76	8.07	8.00	7.89	8.59	9.00	8.67
Kimi-K2	7.03	7.37	6.43	7.71	6.45	8.28	7.55	8.26

其他语言结果 在涵盖其他语言的多语言评估中，GLM-4.5 保持领先，在文本生成和主观问答方面表现出色。

Table 10: Manual Evaluation Scores on Other Language Prompts

Model	Overall	Math	Text Proc.	Text Gen.	Subj. QA	Obj. QA	Code	Logic
GLM-4.5	8.49	8.67	8.13	8.90	9.33	8.71	7.86	8.33
DeepSeek-R1-0528	8.27	9.44	8.38	7.86	9.44	8.22	7.64	8.17
Kimi-K2	6.63	7.22	6.38	7.62	7.78	6.22	6.68	7.17

4.3.2 Evaluation of Coding Agent

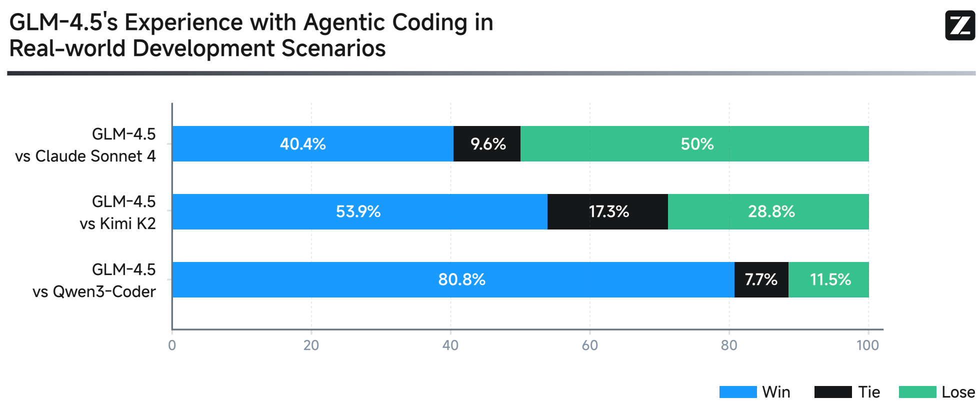

Figure 12: Head-to-head evaluation results between GLM-4.5 and other models on CC-Bench.

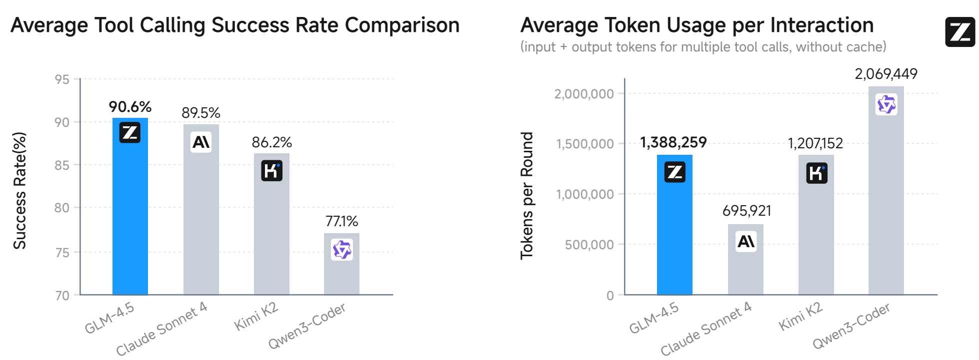

Figure 13: Average tool calling success rate and token usage per interaction across different models on CC-Bench.

Experimental Setup To evaluate the agentic coding capabilities of GLM-4.5 in real-world scenarios, we constructed **CC-Bench**, a benchmark built on the Claude Code[5], encompassing 52 carefully designed programming tasks across diverse software development domains[6]. We compare GLM-4.5 against three strong baselines: Claude Sonnet 4, Kimi K2, and Qwen3-Coder. Each task was executed in an isolated containerized environment to prevent cross-task interference, with models initialized using predefined API configurations. Testing was conducted interactively by human experts over multiple rounds: each task began with a standardized prompt, followed by iterative interactions where

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

[6] The detailed task descriptions and all evaluation trajectories of CC-Bench are available at `https://huggingface.co/datasets/zai-org/CC-Bench-trajectories`

表10：其他语言提示的手动评估分数

Model	Overall	Math	Text Proc.	Text Gen.	Subj. QA	Obj. QA	Code	Logic
GLM-4.5	8.49	8.67	8.13	8.90	9.33	8.71	7.86	8.33
DeepSeek-R1-0528	8.27	9.44	8.38	7.86	9.44	8.22	7.64	8.17
Kimi-K2	6.63	7.22	6.38	7.62	7.78	6.22	6.68	7.17

4.3.2 编码代理的评估

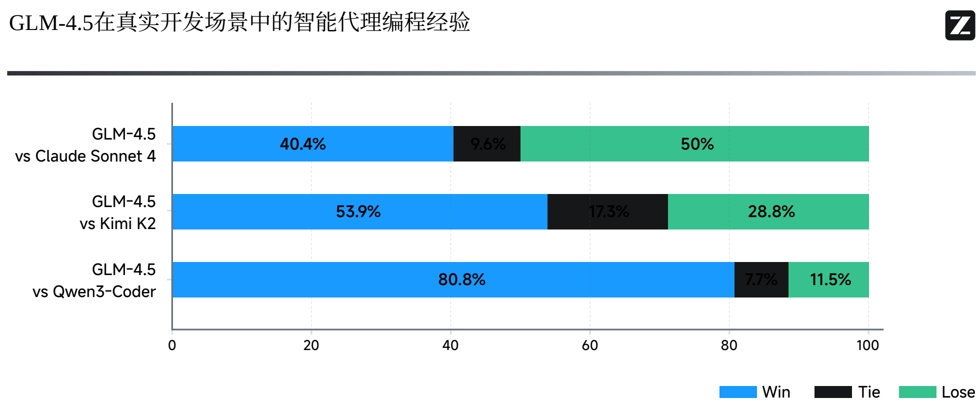

图12：GLM-4.5与其他模型在CC-Bench上的直接评估结果。

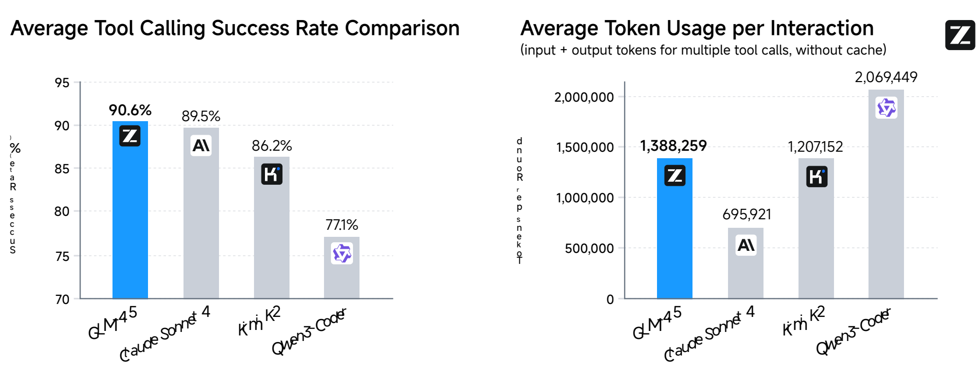

图13：CC-Bench上不同模型的平均工具调用成功率和每次交互的token使用量。

实验设置 为了评估GLM-4.5在真实场景中的智能编程能力，我们构建了CC-Bench，这是一个基于Claude Code[5]构建的基准测试，包含了52个精心设计的编程任务，涵盖了多样化的软件开发领域[6]。我们将GLM-4.5与三个强大的基线模型进行比较：Claude Sonnet 4、Kimi K2和Qwen3-Coder。每个任务都在隔离的容器化环境中执行，以防止跨任务干扰，模型使用预定义的API配置进行初始化。测试由人类专家通过多轮交互进行：每个任务以标准化提示开始，然后是迭代交互，其中

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

[6]The detailed task descriptions and all evaluation trajectories of CC-Bench are available at `https://huggingface.co/datasets/zai-org/CC-Bench-trajectories`

experts adjusted inputs based on model outputs until the task was completed or failed. To ensure fairness, the same expert followed consistent interaction strategies across all models.

Based on this testing procedure, model performance was evaluated using the following criteria: the primary metric was **task completion**, determined by predefined completion criteria. In cases of ties, **efficiency and reliability**—including tool calling success rate and token consumption efficiency—were used as secondary metrics. The evaluation prioritized functional correctness and task completion over efficiency metrics, ensuring that coding capability remained the primary evaluation focus.

Results In head-to-head evaluations, GLM-4.5 demonstrated strong performance relative to open-source baselines and competitive capability against closed-source models as shown in figure 12. Specifically:

- **GLM-4.5 vs Claude Sonnet 4**: 40.4% win, 9.6% tie, 50.0% loss
- **GLM-4.5 vs Kimi K2**: 53.9% win, 17.3% tie, 28.8% loss
- **GLM-4.5 vs Qwen3-Coder**: 80.8% win, 7.7% tie, 11.5% loss

As shown in figure 13, GLM-4.5 particularly excelled in tool calling reliability, achieving the highest success rate at **90.6%**, compared to Claude Sonnet 4 (89.5%), Kimi-K2 (86.2%), and Qwen3-Coder (77.1%). While Claude Sonnet 4 remains a strong competitor, GLM-4.5 outperformed other models in both task completion consistency and agentic execution robustness.

4.3.3 Evaluation of Logical Reasoning

To rigorously assess the true logical reasoning capabilities of the models and mitigate the risk of data contamination from common logical questions found online, we constructed a new, challenging evaluation set. This set comprises novel and complex logical reasoning problems that are structurally different from those widely available on the internet. Each problem is designed to require multiple steps of logical deduction to arrive at the correct solution.

For this evaluation, we established a unified and detailed scoring standard for each question. We then tasked each model with solving these problems. The correctness and quality of each model's response were subsequently inspected and scored by human experts. The results show a competitive landscape, with GLM-4.5 performing on par with leading models.

Table 11: Expert Evaluation Scores on Novel Logical Reasoning Problems

Model	Score
Gemini 2.5 Pro	65.8
DeepSeek-R1-0528	62.1
GLM-4.5	62.0
GLM-4.5-Air	53.4
Kimi K2	51.9

4.4 Evaluation of Translation

The New Paradigm of Translation Translation today extends beyond simple text conversion to encompass a nuanced understanding of evolving internet slang, cultural context, and domain-specific terminology:

Netizen Lingo: Translating "yyds" accurately requires recognizing it as the acronym for the Chinese phrase "永远的神" (yǒng yuǎn de shén), meaning "the eternal god," thus capturing its true sentiment of enthusiastic praise and admiration.

Domain Nicknames: Recognizing "胖" (literally "fat white") is critical within photography communities. Specialized models may translate it incorrectly, but a general-purpose model understands it as a widely used nickname for the "Canon EF 70-300mm f/4-5.6 IS USM" lens, providing precise translations.

专家根据模型输出调整输入，直到任务完成或失败。为确保公平性，同一专家在所有模型中遵循一致的交互策略。

基于此测试流程，模型性能使用以下标准进行评估：主要指标是任务完成情况，由预定义的完成标准确定。在出现平局的情况下，效率和可靠性（包括工具调用成功率和令牌消耗效率）被用作次要指标。评估优先考虑功能正确性和任务完成情况，而非效率指标，确保编码能力仍然是主要的评估重点。

结果 在直接对比评估中，GLM-4.5 相对于开源基线模型表现出强大的性能，并且如图12所示，与闭源模型具有竞争性的能力。具体来说：

- GLM-4.5 对比 Claude Sonnet 4: 40.4% 胜率, 9.6% 平局, 50.0% 败率
- GLM-4.5 vs Kimi K2: 53.9% 胜, 17.3% 平, 28.8% 负
- GLM-4.5 vs Qwen3-Coder: 80.8% 胜, 7.7% 平, 11.5% 负

如图13所示，GLM-4.5在工具调用可靠性方面表现尤为出色，成功率达到90.6%，高于Claude Sonnet 4（89.5%）、Kimi-K2（86.2%）和Qwen3-Coder（77.1%）。虽然Claude Sonnet 4仍然是一个强劲的竞争对手，但GLM-4.5在任务完成一致性和代理执行稳健性方面均优于其他模型。

4.3.3 逻辑推理的评估

为了严格评估模型的真实逻辑推理能力，并降低来自网络上常见逻辑问题的数据污染风险，我们构建了一个新的、具有挑战性的评估集。该评估集包含新颖且复杂的逻辑推理问题，这些问题在结构上与互联网上广泛可用的逻辑问题不同。每个问题都设计为需要多步骤的逻辑推理才能得出正确答案。

在这项评估中，我们为每个问题建立了统一和详细的评分标准。然后我们要求每个模型解决这些问题。每个模型的回答的正确性和质量随后由人类专家进行检查和评分。结果显示了一个竞争格局，GLM-4.5的表现与领先模型相当。

表11：专家对新颖逻辑推理问题的评估分数

Model	**Score**
Gemini 2.5 Pro	65.8
DeepSeek-R1-0528	62.1
GLM-4.5	62.0
GLM-4.5-Air	53.4
Kimi K2	51.9

4.4 翻译评估

翻译的新范式　如今的翻译已超越简单的文本转换，扩展到对不断发展的网络俚语、文化背景和领域特定术语的细致理解：

Netizen Lingo: 准确翻译"yyds"需要认识到它是中文短语"永远的神" (yǒng yuǎn de shén)的首字母缩写，意为"永恒的神"，从而捕捉到其表达的热烈赞美和钦佩的真实情感。

Domain Nicknames: 在摄影社区中，识别"胖"（字面意思是"胖白"）至关重要。专业模型可能会错误地翻译它，但通用模型将其理解为"Canon EF 70-300mm f/4-5.6 IS USM"镜头的常用昵称，从而提供准确的翻译。

Symbols: When a Chinese user sends a "fish" emoji in a conversation to refer to a second-hand marketplace, can the model understand the cultural meme behind it, which points to the "闲鱼" (Xiányú) platform? This tests the model's cognitive ability to connect visual symbols with online cultural phenomena.

Deep Contextual Reasoning: Translating "三花公主驾到，速来围观" demands identifying "三花" not as a person's name but as a reference to the popular calico coloration of cats. A general-purpose model accurately deduces this context, translating the phrase idiomatically as "The Calico Princess has arrived! Come and see!".

These examples underscore modern translation as a task rooted deeply in knowledge and reasoning.

Evaluation Results We tested 100 challenging, real-world cases commonly mistranslated by current tools, comparing GLM-4.5 against specialized translation models (Qwen-MT-plus, Qwen-MT-turbo, Seed-X [9]) in a blind human evaluation (scored 0-3 considering whether the meaning is conveyed correctly and whether the language is authentic). The results are shown in Table 12.

Table 12: Human Scores on Selected Challenging Translation data

Model	Average Score
GLM-4.5	**1.71**
Qwen-MT-plus	0.38
Qwen-MT-turbo	0.55
Seed-X	0.65

GLM-4.5 significantly outperforms specialized models. For example, translating "三花公主驾到" specialized models failed contextually, whereas GLM-4.5 accurately conveys the idiomatic meaning.

5 Conclusion

In this report, we have introduced the GLM-4.5 model series, including GLM-4.5 and GLM-4.5-Air. Both models adopt the MoE architecture, which improves the computational efficiency compared to previous GLM models. GLM-4.5 excels at reasoning, coding, and agentic tasks, ranked in 3rd place globally among open-source and proprietary models. We release the model weights of GLM-4.5 and GLM-4.5-Air to advance the applications and research of large language models.

*Symbols*当中国用户在对话中发送"鱼"表情符号来指代二手市场时，模型能否理解其背后的文化迷因，该迷因指向"闲鱼"(Xiányú)平台？这测试了模型将视觉符号与网络文化现象联系起来的认知能力。

Deep Contextual Reasoning: 翻译"三花公主驾到，速来围观"需要识别"三花"不是人名，而是指猫咪流行的三花猫毛色。通用模型能够准确推断这一语境，将短语习语化地翻译为"三花公主驾到！快来围观！"。

这些例子强调了现代翻译是一项深深植根于知识和推理的任务。

评估结果 我们测试了100个具有挑战性的真实案例，这些案例通常被当前工具错误翻译，在盲人评估中，将GLM-4.5与专业翻译模型（Qwen-MT-plus、Qwen-MT-turbo、Seed-X [9]）进行比较（评分0-3，考虑是否正确传达了意思以及语言是否地道）。结果如表12所示。

表12：关于选定挑战性翻译的人类评分 数据

Model	Average Score
GLM-4.5	**1.71**
Qwen-MT-plus	0.38
Qwen-MT-turbo	0.55
Seed-X	0.65

GLM-4.5 的表现显著优于专业模型。例如，在翻译"三花公主驾到"时，专业模型在语境理解上失败了，而 GLM-4.5 则准确传达了其习语含义。

5 结论

在本报告中，我们介绍了GLM-4.5模型系列，包括GLM-4.5和GLM-4.5-Air。这两个模型都采用了MoE架构，与之前的GLM模型相比提高了计算效率。GLM-4.5在推理、编程和智能体任务方面表现出色，在开源和专有模型中全球排名第三。我们发布GLM-4.5和GLM-4.5-Air的模型权重，以促进大语言模型的应用和研究。

6 Contribution

Contributors' names are listed in alphabetical order by first name. Names marked with an asterisk (*) indicate individuals who have since left our team.

Core Contributors
Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang (汪子涵), Zilin Zhu

Contributors
Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu*, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li*, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang*, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang*, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang*, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai*, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li (李泷), Shuang Li (李爽), Shuxun Yang, Sibo Yi*, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang*, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang*, Yuhang Li, Yuhao Wu*, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang (王梓汉), Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao*

Tech Leads
Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou

Advisors
Jie Tang, Yuxiao Dong, Juanzi Li, Hongning Wang, Minlie Huang, Bin Xu, Jidong Zhai, Wenguang Chen

Acknowledgement
We are grateful for all the support from Beijing, Shanghai, Tianjin, Hangzhou, Zhuhai, and Chengdu. Special thanks to our customers and community developers.

6 贡献

贡献者的名字按名字首字母顺序排列。标有星号(*)的名字表示已经离开我们团队的成员。

核心贡献者
陈斌, 谢程星, 王存祥, 尹达, 曾浩, 张佳杰, 王克东, 钟鲁宸, 刘明道, 卢瑞, 曹树霖, 张晓涵, 黄宣成, 魏尧, 程彦, 安逸凡, 牛艺霖, 温源昊, 白宇石, 杜正潇, Zihan Wang (汪子涵), 朱子林

贡献者
张博涵，闻思博，吴博文，徐博文*，黄灿，赵凯西，蔡长鹏，于超，李辰，葛晨迪，黄程华，张晨辉，徐晨曦，朱辰正，李创*，封丛峰，林道言，杨大勇，蒋大志，丁爱，朱二乐，王飞，盛正，王国，孙海龙，李海涛，李海洋，胡海逸，张涵宇，彭浩，浩泰，张浩可，王浩然，杨浩宇*，何刘，何赵，刘宏伟，闫希红，刘欢，陈慧龙，李季，赵佳静，任佳敏，焦健，赵佳妮，闫建阳，王佳琪*，桂佳怡，赵佳月，刘杰，李继杰，李静，卢静，王景森，袁静伟，李静轩，杜静昭，杜金华，刘金鑫，支俊凯，高俊丽，王珂，杨乐康*，徐亮，范林，吴林东，丁林涛，王璐，张曼，李明浩，徐明欢，赵明明，翟明舒*，杜鹏凡，董倩，雷尚德，涂尚清，杨尚通，卢少友，李世杰，李泷，李爽，杨树勋，伊思博*，于天舒，田伟，王维涵，于文博，谭文林，梁文杰，刘文涛，王晓*，贾晓涵，顾小涛，凌晓英，王欣，范星，潘星如，张欣源，张欣泽，傅秀清，张勋凯，徐亚博，吴彦东，卢一达，王一东，周艺林，潘一鸣，张颖，王英丽，李英茹，苏银培，耿一鹏，朱一彤，杨永坤*，李宇航，吴宇航*，李江宇，刘宇楠，王云清，李云涛，张宇轩，刘泽震，杨震，周正大，乔培忠，冯卓尔，刘若卓，张子宸，王梓汉，姚子君，王子康，刘自强，柴子薇，李子轩，赵佐东*

技术主管
曾奥瀚, 吕欣, 郑楷, 侯宇

顾问
唐杰, 董宇晓, 李娟子, 王红宁, 黄民烈, 徐斌, 纪东, 陈广文

致谢
我们感谢来自北京、上海、天津、杭州、珠海和成都的所有支持.特别感谢我们的客户和社区开发者.

References

[1] A. Abbas, K. Tirumala, D. Simig, S. Ganguli, and A. S. Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

[2] C. An, Z. Xie, X. Li, L. Li, J. Zhang, S. Gong, M. Zhong, J. Xu, X. Qiu, M. Wang, and L. Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025.

[3] Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, 2024.

[4] Y. Bai, S. Tu, J. Zhang, H. Peng, X. Wang, X. Lv, S. Cao, J. Xu, L. Hou, Y. Dong, J. Tang, and J. Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, Vienna, Austria, July 2025. Association for Computational Linguistics.

[5] M. Bavarian, H. Jun, N. Tezak, J. Schulman, C. McLeavey, J. Tworek, and M. Chen. Efficient training of language models to fill in the middle, 2022.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] A. Chen, A. Li, B. Gong, B. Jiang, B. Fei, B. Yang, B. Shan, C. Yu, C. Wang, C. Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

[8] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[9] S. Cheng, Y. Bao, Q. Cao, L. Huang, L. Kang, Z. Liu, Y. Lu, W. Zhu, Z. Huang, T. Li, et al. Seed-x: Building strong multilingual translation llm with 7b parameters. *arXiv preprint arXiv:2507.13618*, 2025.

[10] K. Deshpande, V. Sirdeshmukh, J. B. Mols, L. Jin, E.-Y. Hernandez-Cardona, D. Lee, J. Kritz, W. E. Primack, S. Yue, and C. Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18632–18702, 2025.

[11] H. Ding, Z. Wang, G. Paolini, V. Kumar, A. Deoras, D. Roth, and S. Soatto. Fewer truncations improve language modeling. In *Proceedings of the 41st International Conference on Machine Learning*, pages 11030–11048, 2024.

[12] F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

[13] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[15] A. Henry, P. R. Dachapally, S. Pawar, and Y. Chen. Query-key normalization for transformers, 2020.

[16] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, and B. Ginsburg. Ruler: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

参考文献

[1] A. Abbas, K. Tirumala, D. Simig, S. Ganguli, 和 A. S. Morcos. Semdedup: 通过语义去实现网络规模的高效数据学习. *arXiv preprint arXiv:2303.09540*, 2023. [2] C. An, Z. Xie, X. Li, L. Li, J. Zhang, S. Gong, M. Zhong, J. Xu, X. Qiu, M. Wang, 和 L. Kong. Polaris: 针对高级推理模型扩展强化学习的后训练方法, 2025. [3] Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, 等. Longbench: 一个用于长上下文理解的双语、多任务基准测试. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, 2024. [4] Y. Bai, S. Tu, J. Zhang, H. Peng, X. Wang, X. Lv, S. Cao, J. Xu, L. Hou, Y. Dong, J. Tang, 和 J. Li. LongBench v2: 面向真实长上下文多任务的更深层次理解和推理. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, 奥地利维也纳, 2025年7月. 计算语言学协会. [5] M. Bavarian, H. Jun, N. Tezak, J. Schulman, C. McLeavey, J. Tworek, 和 M. Chen. 高效训练语言模型以填补中间内容, 2022.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. 语言模型是少样本学习者。 *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] A. Chen, A. Li, B. Gong, B. Jiang, B. Fei, B. Yang, B. Shan, C. Yu, C. Wang, C. Zhu, 等. Minimax-m1: 通过闪电注意力高效扩展测试时计算. *arXiv preprint arXiv:2506.13585*, 2025.

[8] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. 评估在代码上训练的大型语言模型。*arXiv preprint arXiv:2107.03374*, 2021.

[9] S. Cheng, Y. Bao, Q. Cao, L. Huang, L. Kang, Z. Liu, Y. Lu, W. Zhu, Z. Huang, T. Li, et al. Seed-x: 构建具有70亿参数的强大多语言翻译大语言模型。*arXiv preprint arXiv:2507.13618*, 2025.

[10] K. Deshpande, V. Sirdeshmukh, J. B. Mols, L. Jin, E.-Y. Hernandez-Cardona, D. Lee, J. Kritz, W. E. Primack, S. Yue, and C. Xing. Multichallenge：一个对前沿大型语言模型具有挑战性的现实多轮对话评估基准. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18632–18702, 2025.

[11] H. Ding, Z. Wang, G. Paolini, V. Kumar, A. Deoras, D. Roth, and S. Soatto. 更少的截断改进了语言建模。在 *Proceedings of the 41st International Conference on Machine Learning* 中，第 11030-11048 页，2024年。

[12] F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. 通过多令牌预测实现更好更快的大型语言模型。*arXiv preprint arXiv:2404.19737*, 2024. [13] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: 通过强化学习激励大型语言模型的推理能力。*arXiv preprint arXiv:2501.12948*, 2025. [14] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. 使用数学数据集衡量数学问题解决能力。在 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)* 中。[15] A. Henry, P. R. Dachapally, S. Pawar, and Y. Chen. 针对Transformer的查询-键归一化，2020年。[16] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, and B. Ginsburg. Ruler: 您的长上下文语言模型的真实上下文大小是多少？在 *First Conference on Language Modeling* 中。

[17] S. Hu, Y. Tu, X. Han, G. Cui, C. He, W. Zhao, X. Long, Z. Zheng, Y. Fang, Y. Huang, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*.

[18] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[19] N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*.

[20] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[21] K. Jordan, Y. Jin, V. Boza, Y. Jiacheng, F. Cecista, L. Newhouse, and J. Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan.github.io/posts/muon*, 6.

[22] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

[23] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[24] J. Liu, J. Su, X. Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

[25] M. Luo, S. Tan, J. Wong, X. Shi, W. Y. Tang, M. Roongta, C. Cai, J. Luo, L. E. Li, R. A. Popa, and I. Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

[26] S. G. Patil, H. Mao, C. Cheng-Jie Ji, F. Yan, V. Suresh, I. Stoica, and J. E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[27] G. Penedo, H. Kydlíček, V. Sabolčec, B. Messmer, N. Foroutan, A. H. Kargaran, C. Raffel, M. Jaggi, L. Von Werra, and T. Wolf. Fineweb2: One pipeline to scale them all–adapting pre-training data processing to every language. *arXiv preprint arXiv:2506.20920*, 2025.

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

[29] Y. Qin, T. Zhang, Y. Shen, W. Luo, Y. Zhang, Y. Qiao, Z. Zhou, W. Zhang, B. CUI, et al. Sysbench: Can llms follow system message? In *The Thirteenth International Conference on Learning Representations*, 2024.

[30] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[31] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[32] D. Su, K. Kong, Y. Lin, J. Jennings, B. Norick, M. Kliegl, M. Patwary, M. Shoeybi, and B. Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. *arXiv preprint arXiv:2412.02595*, 2024.

[33] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[17] S. Hu, Y. Tu, X. Han, G. Cui, C. He, W. Zhao, X. Long, Z. Zheng, Y. Fang, Y. Huang, et al. Minicpm：通过可扩展的训练策略释放小型语言模型的潜力。In *First Conference on Language Modeling*.

[18] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, 等. Openai o1 系统卡片. *arXiv preprint arXiv:2412.16720*, 2024.

[19] N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, 和 I. Stoica. Livecodebench: 对大型语言模型进行代码的整体和无污染评估。在 *The Thirteenth International Conference on Learning Representations* 中。[20] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, 和 K. Narasimhan. Swe-bench: 语言模型能否解决真实的 GitHub 问题? *arXiv preprint arXiv:2310.06770*, 2023。[21] K. Jordan, Y. Jin, V. Boza, Y. Jiacheng, F. Cecista, L. Newhouse, 和 J. Bernstein. Muon: 神经网络隐藏层的优化器，2024。*URL https://kellerjordan.github.io/posts/muon*, 6。[22] A. Joulin, E. Grave, P. Bojanowski, 和 T. Mikolov. 高效文本分类的技巧集。在 *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* 中，第 427-431 页。计算语言学协会，2017年4月。[23] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, 等。Deepseek-v3 技术报告。*arXiv preprint arXiv:2412.19437*, 2024。[24] J. Liu, J. Su, X. Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, 等。Muon 对 LLM 训练具有可扩展性。*arXiv preprint arXiv:2502.16982*, 2025。

[25] M. Luo, S. Tan, J. Wong, X. Shi, W. Y. Tang, M. Roongta, C. Cai, J. Luo, L. E. Li, R. A. Popa, and I. Stoica. Deepscaler: 通过扩展RL超越o1-preview的1.5B模型. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model- by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion 博客.

[26] S. G. Patil, H. Mao, C. Cheng-Jie Ji, F. Yan, V. Suresh, I. Stoica, and J. E. Gonzalez. 伯克利函数调用排行榜 (bfcl): 从工具使用到大型语言模型的智能体评估。在 *Forty-second International Conference on Machine Learning*, 2025.

[27] G. Penedo, H. Kydlíček, V. Sabolčec, B. Messmer, N. Foroutan, A. H. Kargaran, C. Raffel, M. Jaggi, L. Von Werra, and T. Wolf. Fineweb2: 一条管道扩展全部——将预训练数据处理适应所有语言。*arXiv preprint arXiv:2506.20920*, 2025.

[28] L. Phan, A. Gatti, Z. Han, N. Li, J. Hu, H. Zhang, C. B. C. Zhang, M. Shaaban, J. Ling, S. Shi, et al. 人类的最后一次考试. *arXiv preprint arXiv:2501.14249*, 2025.

[29] Y. Qin, T. Zhang, Y. Shen, W. Luo, Y. Zhang, Y. Qiao, Z. Zhou, W. Zhang, B. CUI, et al. Sysbench：大语言模型能否遵循系统消息？在 *The Thirteenth International Conference on Learning Representations*, 2024.

[30] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, 和 S. R. Bowman. Gpqa: 一个研究生级别的谷歌无法回答的问答基准测试. In *First Conference on Language Modeling*, 2024.

[31] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: 推动开放语言模型中数学推理的极限。 *arXiv preprint arXiv:2402.03300*, 2024.

[32] D. Su, K. Kong, Y. Lin, J. Jennings, B. Norick, M. Kliegl, M. Patwary, M. Shoeybi, and B. Catanzaro. Nemotron-cc: 将Common Crawl转化为一个精细的长视野预训练数据集. *arXiv preprint arXiv:2412.02595*, 2024.

[33] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. Gemini: 一系列高度能用的多模态模型。*arXiv preprint arXiv:2312.11805*, 2023.

[34] K. Team, Y. Bai, Y. Bao, G. Chen, J. Chen, N. Chen, R. Chen, Y. Chen, Y. Chen, Y. Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[35] T. T.-B. Team. Terminal-bench: A benchmark for ai agents in terminal environments, Apr 2025.

[36] M. Tian, L. Gao, S. Zhang, X. Chen, C. Fan, X. Guo, R. Haas, P. Ji, K. Krongchon, Y. Li, et al. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024.

[37] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[38] K. Vodrahalli, S. Ontanon, N. Tripuraneni, K. Xu, S. Jain, R. Shivanna, J. Hui, N. Dikkala, M. Kazemi, B. Fatemi, R. Anil, E. Dyer, S. Shakeri, R. Vij, H. Mehta, V. Ramasesh, Q. Le, E. Chi, Y. Lu, O. Firat, A. Lazaridou, J.-B. Lespiau, N. Attaluri, and K. Olszewska. Michelangelo: Long context evaluations beyond haystacks via latent structure queries, 2024.

[39] F. Wan, W. Shen, S. Liao, Y. Shi, C. Li, Z. Yang, J. Zhang, F. Huang, J. Zhou, and M. Yan. Qwenlong-l1: Towards long-context large reasoning models with reinforcement learning. *arXiv preprint arXiv:2505.17667*, 2025.

[40] L. Wang, H. Gao, C. Zhao, X. Sun, and D. Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

[41] S. Wang, L. Yu, C. Gao, C. Zheng, S. Liu, R. Lu, K. Dang, X. Chen, J. Yang, Z. Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

[42] X. Wang, B. Li, Y. Song, F. F. Xu, X. Tang, M. Zhuge, J. Pan, Y. Song, B. Li, J. Singh, H. H. Tran, F. Li, R. Ma, M. Zheng, B. Qian, Y. Shao, N. Muennighoff, Y. Zhang, B. Hui, J. Lin, R. Brennan, H. Peng, H. Ji, and G. Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

[43] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[44] J. Wei, N. Karina, H. W. Chung, Y. J. Jiao, S. Papay, A. Glaese, J. Schulman, and W. Fedus. Measuring short-form factuality in large language models, 2024.

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

[46] Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, D. Yang, C. Liao, X. Guo, W. He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

[47] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[48] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. tau-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

[49] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, W. Dai, T. Fan, G. Liu, L. Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[50] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, et al. Glm-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*.

[34] K. 团队, Y. 白, Y. 鲍, G. 陈, J. 陈, N. 陈, R. 陈, Y. 陈, Y. 陈, Y. 陈, 等. Kimi k2: 开放智能代理. *arXiv preprint arXiv:2507.20534*, 2025. [35] T. T.-B. 团队. Terminal-bench: 终端环境中AI代理的基准测试, 2025年4月. [36] M. 田, L. 高, S. 张, X. 陈, C. 范, X. 郭, R. 哈斯, P. 季, K. Krongchon, Y. 李, 等. Scicode: 科学家策划的研究编码基准. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024. [37] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, 等. Llama: 开放高效的基础语言模型. *arXiv preprint arXiv:2302.13971*, 2023. [38] K. Vodrahalli, S. Ontanon, N. Tripuraneni, K. Xu, S. Jain, R. Shivanna, J. Hui, N. Dikkala, M. Kazemi, B. Fatemi, R. Anil, E. Dyer, S. Shakeri, R. Vij, H. Mehta, V. Ramasesh, Q. Le, E. Chi, Y. Lu, O. Firat, A. Lazaridou, J.-B. Lespiau, N. Attaluri, 和 K. Olszewska. Michelangelo: 通过潜在结构查询超越干草堆的长上下文评估, 2024.

[39] F. Wan, W. Shen, S. Liao, Y. Shi, C. Li, Z. Yang, J. Zhang, F. Huang, J. Zhou, and M. Yan. Qwenlong-l1: 朝向具有强化学习的长上下文大推理模型. *arXiv preprint arXiv:2505.17667*, 2025.

[40] L. Wang, H. Gao, C. Zhao, X. Sun, and D. Dai. 用于专家混合模型的无辅助损失负载均衡策略。*arXiv preprint arXiv:2408.15664*, 2024.

[41] S. Wang, L. Yu, C. Gao, C. Zheng, S. Liu, R. Lu, K. Dang, X. Chen, J. Yang, Z. Zhang, et al. 超越二八定律：高熵少数令牌驱动大语言模型推理的有效强化学习。*arXiv preprint arXiv:2506.01939*, 2025.

[42] X. Wang, B. Li, Y. Song, F. F. Xu, X. Tang, M. Zhuge, J. Pan, Y. Song, B. Li, J. Singh, H. H. Tran, F. Li, R. Ma, M. Zheng, B. Qian, Y. Shao, N. Muennighoff, Y. Zhang, B. Hui, J. Lin, R. Brennan, H. Peng, H. Ji, and G. Neubig. Openhands: 面向AI软件开发者的通用智能体开放平台. In *The Thirteenth International Conference on Learning Representations*, 2025.

[43] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, et al. Mmlu-pro: 一个更稳健且具有挑战性的多任务语言理解基准. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[44] J. Wei, N. Karina, H. W. Chung, Y. J. Jiao, S. Papay, A. Glaese, J. Schulman, and W. Fedus. 测量大型语言模型中的短形式事实性, 2024.

[45] J. Wei, Z. Sun, S. Papay, S. McKinney, J. Han, I. Fulford, H. W. Chung, A. T. Passos, W. Fedus, and A. Glaese. Browsecomp: 一个简单却具有挑战性的浏览代理基准测试。*arXiv preprint arXiv:2504.12516*, 2025.

[46] Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, D. Yang, C. Liao, X. Guo, W. He, et al. Agentgym: 在多样化环境中进化基于大型语言模型的智能体。*arXiv preprint arXiv:2406.04151*, 2024.

[47] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, 等. Qwen3 技术报告. *arXiv preprint arXiv:2505.09388*, 2025. [48] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. tau-bench: 面向现实领域中工具-代理-用户交互的基准测试. *arXiv preprint arXiv:2406.12045*, 2024. [49] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, W. Dai, T. Fan, G. Liu, L. Liu, 等. Dapo: 一个大规模开源大语言模型强化学习系统. *arXiv preprint arXiv:2503.14476*, 2025.

[50] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, 等. Glm-130b: 一个开放的双语预训练模型. In *The Eleventh International Conference on Learning Representations*.

[51] Z. Zhang, L. Lei, L. Wu, R. Sun, Y. Huang, C. Long, X. Liu, X. Lei, J. Tang, and M. Huang. Safetybench: Evaluating the safety of large language models with multiple choice questions. *arXiv preprint arXiv:2309.07045*, 2023.

[52] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[51] Z. Zhang, L. Lei, L. Wu, R. Sun, Y. Huang, C. Long, X. Liu, X. Lei, J. Tang, and M. Huang. Safetybench: 使用多项选择题评估大型语言模型的安全性. *arXiv preprint arXiv:2309.07045*, 2023. [52] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. 大型语言模型的指令遵循评估. *arXiv preprint arXiv:2311.07911*, 2023.