TeleChat2、TeleChat2.5 和 T1 的技术报告

Zihan Wang*, Xinzhang Liu*, Yitong Yao*, Chao Wang*, Yu Zhao*, Zhihao Yang*, Wenmin Deng*, Kaipeng Jia, Jiaxin Peng, Yuyao Huang, Sishi Xiong, Zhuo Jiang, Kaidong Yu, Xiaohui Hu, Fubei Yao, Ruiyu Fang, Zhuoru Jiang, Ruiting Song, Qiyi Xie, Rui Xue, Xuewei He, Yanlei Xue, Zhu Yuan, Zhaoxi Zhang, Zilu Huang, Shiquan Wang, Xin Wang, Hanming Wu, Mingyuan Wang, Xufeng Zhan, Yuhan Sun, Zhaohu Xing, Yuhao Jiang, Bingkai Yang, Shuangyong Song, Yongxiang Li, Zhongjiang He†, Xuelong Li†

TeleAI

Abstract

我们推出了最新系列的 TeleChat 模型：TeleChat2、TeleChat2.5 和 T1，相较于其前身 TeleChat 进行了重要升级。尽管模型架构变化不大，但该系列通过在预训练和后训练阶段增强的训练策略，实现了显著的性能提升。该系列首先推出的是 TeleChat2，其在 10 万亿高质量和多样性的 tokens 上进行预训练。接下来是监督微调（SFT）和直接偏好优化（DPO）以进一步增强其能力。TeleChat2.5 和 T1 扩展了流程，通过结合使用特定领域的数据集进行持续预训练阶段以及强化学习（RL）来提升代码生成和数学推理任务的性能。T1 变体专为复杂推理而设计，支持长链式思考（CoT）推理，并在数学和编码方面显示出显著改进。相反，TeleChat2.5 优先考虑速度，提供快速推理。这两款旗舰模型 T1 和 TeleChat2.5 都是基于 115B 参数密集型 Transformer 架构，与原始 TeleChat 相比，在推理和通用任务性能上表现出显著进步。值得注意的是，T1-115B 在性能上超过了 OpenAI 的 o1-mini 和 GPT-4o 等专有模型。我们公开发布 TeleChat2、TeleChat2.5 和 T1，包括后训练版本的 35B 和 115B 参数，旨在为开发人员和研究人员提供最先进的、适用于多种应用的语言模型。

Model	Link
TeleChat2-35B	https://modelscope.cn/models/TeleAI/TeleChat2-35B
TeleChat2-115B	https://modelscope.cn/models/TeleAI/TeleChat2-115B
TeleChat2.5-35B	https://modelscope.cn/models/TeleAI/TeleChat2.5-35B
TeleChat2.5-115B	https://modelscope.cn/models/TeleAI/TeleChat2.5-115B
T1-35B	https://modelscope.cn/models/TeleAI/T1-35B
T1-115B	https://modelscope.cn/models/TeleAI/T1-115B

Model Series	Github Link
TeleChat2	https://github.com/Tele-AI/TeleChat2
TeleChat2.5	https://github.com/Tele-AI/TeleChat2.5
T1	https://github.com/Tele-AI/T1

*These authors contributed equally to this work
†Correspondence to hezj@chinatelecom.cn; xuelong_li@chinatelecom.cn

Contents

近年来，大型语言模型（LLMs）迅速发展和增长，标志着在通用人工智能（AGI）方面的进步。诸如 GPT4、Claude 和 Gemini 等专有产品展示了与人类能力相当的表现，提高了社区对开源 LLMs 潜力的期望值。除了专有模型外，还有几个值得注意的开源 LLMs，例如 LLaMA 系列、Qwen 系列、Mistral 系列、Deepseek 系列，以及我们的 TeleChat 系列也在取得显著进展，努力缩小与其专有对应的差距。开放权重模型使大型语言模型对开发者更加可及，促进了更广泛的研究参与，通过社区合作推动创新，加速了跨各个领域的人工智能应用的发展。最近的进展，例如 DeepSeek-R1 的成功，展示了长推理链（COT）和强化学习（RL）在增强大型语言模型（LLMs）推理能力中的关键作用。引人注目的例子，包括 OpenAI-o1、Skywork OR1、Qwen3、和 Kimi-K1.5，说明了在复杂任务如数学推理和代码生成中，RL 能够显著提高性能。

为了推进开源贡献，我们增强了我们的模型，并推出了最新系列，包括 TeleChat2、TeleChat2.5 和 T1，相较于其前身 TeleChat 进行了显著升级。这些开源权重的发布包括经过后期训练的 35B 和 115B 参数语言模型的变体。我们已经在 HuggingFace 和 Mod-

elScope 等平台上公开了模型参数。此外，我们在 GitHub 上提供了完整的代码库，包括用于模型微调、量化、部署以及与 LangChain 集成的全面工具，从而支持广泛的实际应用。

新系列 TeleChat 的发展包括两个主要阶段：(1) 预训练阶段，其中模型通过预测连续文本中的下一个单词在海量数据集上进行训练。TeleChat2 的预训练过程被详细描述，突出了多种数据类型的准备和数据组成的调整。TeleChat2 能够有效捕捉长期依赖关系，最初在 8K 令牌上进行训练，然后在预训练阶段推进到 256K 令牌，在长文本基准测试中表现出显著的性能。(2) 随后，我们进行后训练，包括在 TeleChat2 基础模型上的监督微调（SFT, Ouyang et al. (2022)）和直接偏好优化（DPO, Rafailov et al. (2023)），以使其符合人类偏好并进一步提高特定能力。本文还提供了我们在增强特定能力方面的见解，例如代码、推理、工具使用和精确指令跟随。

TeleChat2、TeleChat2.5 和 T1 系列是在由 8,000 个 Ascend NPU 提供支持的 Atlas 900 A2 集群上训练的。分布式训练利用了 MindSpore 的大型模型并行框架 [1] 提供的 4D 并行策略，实现了万亿参数模型的高效扩展。计算基础设施由 CTYun 在上海计算中心托管，提供了大规模训练所需的高性能资源。

新的系列旨在增强模型理解和生成自然语言文本的能力，特别是在复杂和细微的语境中。为了评估它们在这些场景中的表现，我们在数学、推理、工具使用、精确指令遵循和开放式任务等综合基准集上测试了 TeleChat2、TeleChat2.5 和 T1。评估结果表明，这些模型在推理和一般任务性能上相比它们的前代 TeleChat 取得了显著进步。值得注意的是，T1-115B 的表现优于诸如 OpenAI’s o1-mini 和 GPT-4o 等专有模型。

- 在训练数据中表现更好。我们提高了用于预训练和后训练的数据的质量和数量。在预训练阶段，我们将高质量预训练数据集从之前的 3 万亿标记扩展到 10 万亿标记，为常识、专业知识和推理能力奠定了坚实的基础。至于后训练数据，我们采用更严格的质量控制和过滤流程，并精心收集高质量数据以增强几个特定的能力。此外，我们进行了数据融合实验，以识别预训练和后训练阶段的最佳数据组合。
- 模型规模更大。相比于 TeleChat 系列的以往版本，我们在一个显著更大规模上开发了新的模型系列。我们旗舰产品 TeleChat2、TeleChat2.5 和 T1 拥有 1150 亿可训练参数。此外，我们还推出了 35B 变体，为资源受限的场景提供了更具成本效益的解决方案。鉴于新系列模型共享统一的模型架构，并在同质源数据上进行训练，但大小不同，它们可以在 AI-Flow 框架 (Shao & Li, 2024) (An et al., 2025) 下协同工作，将工作负载分布在不同计算节点中的多个模型上，包括终端设备、边缘节点和云服务器。这促进了跨网络的智能无缝流动。
- 在实际应用中表现更佳。新的模型系列经过训练，显著扩展了模型的上下文长度，超出了 TeleChat 的上下文长度，支持高达 128K 个标记的上下文窗口。这一增强对于实际应用至关重要，例如冗长的对话、长距离推理和理解，以及其他需要模型考虑大量上下文的任务。此外，新的系列还提供了更好且更易于使用的工具，使其对于广泛的应用更加方便和用户友好。
- 在推理能力和编码方面更优秀。TeleChat2.5 和 T1 的训练结合了强化学习（RL），以明确优化模型解决数学和编码问题的能力，与其前身相比，在处理这些领域的复杂问题时显示出了显著的改进。

在本文的其余部分中，我们首先在第 1 节介绍模型架构。接下来，我们在第 2 节描述我们的预训练过程，包括预训练方案、训练数据的构建以及长上下文扩展技术。之后，我们讨论我们的后训练方法，包括监督微调（第 2.2 节）、方向偏好优化（第 2.3 节）和强化学习（第 2.4 节）期间训练数据的组成和具体方法。我们在第 ?? 节突出介绍了对某些特定能力（如代码、数学推理、工具使用和精确指令遵循）性能改进的特殊努力。我们描述了支持训练的硬件和基础设施，并讨论了在第 ?? 节中导致训练效率提升的几种优化。然后，我们在第 3 节中呈现详细的评估结果，涵盖基础模型和聊天模型。

1 模型架构

TeleChat2、TeleChat2.5 和 T1 采用统一的模型架构，大致保留了其前身 TeleChat 的设计。该架构采用了带有 RMSNorm 归一化的 Pre-Norm 设计 (Zhang & Sennrich, 2019)，在前馈网络（FFN）中使用 SwiGLU (Shazeer, 2020) 作为激活函数，并整合了旋转位置嵌入 (Su

[1] https://www.mindspore.cn/

et al., 2022) 。详细的网络规格可以在表 1 中找到。与 TeleChat 相比有几个小的调整，具体如下：

- 分组查询注意力 (GQA)。我们使用具有 8 个键值头的分组查询注意力来替代传统的多头注意力 (MHA)，用于拥有 1150 亿参数的模型，实现了加速训练和高效的 KV 缓存利用。
- RoPE 基本频率。通过增加 RoPE 基本频率超参数，我们提高了更有效处理更长上下文的能力。详见第 ?? 节。

$Params$	n_{layers}	d_{model}	d_{ffn}	n_{heads}	n_{kv_heads}	n_{vocab}
35B	64	6144	20480	48	48	131072
115B	96	8192	40960	64	8	131072

Table 1: TeleChat2、TeleChat2.5 和 T1 模型系列的详细模型架构超参数。

2 预训练

我们的 TeleBase2 训练过程包括两个主要阶段。首先，在初始预训练阶段（第 2.1 节），我们通过过滤和数据混合来策划高质量的训练数据，总共得到 10 万亿个标记。在这一阶段，模型掌握基础的语言结构，并从文本数据中积累广泛的世界知识。其次，在长上下文退火阶段（第 ?? 节），我们通过精心挑选和合成的数据集来优化模型的能力，特别是在推理和基于知识的任务上的性能。同时，我们将模型的上下文长度扩展到 256K 个标记。在后续部分中，我们将从数据组成和训练方法的角度详细阐述这些阶段。预训练框架如图 1 所示。

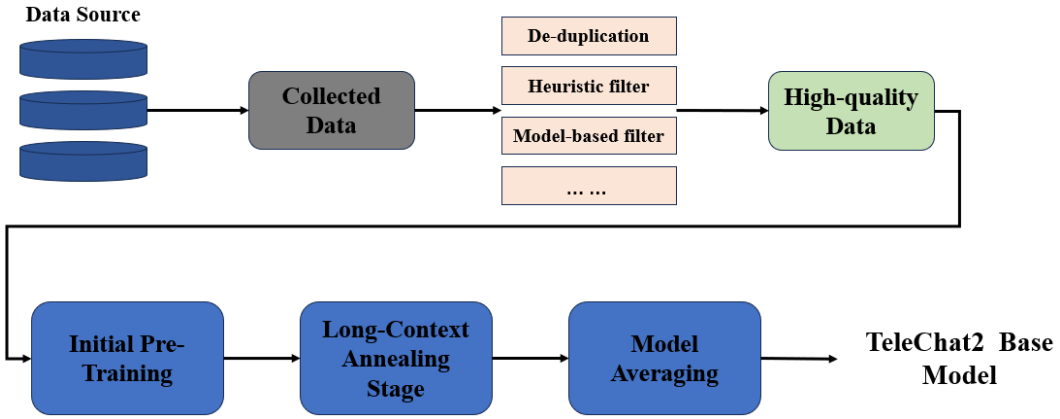

Figure 1: 预训练框架。

2.1 初始预训练阶段

在初始预训练阶段，我们的主要目标是使模型具备广泛和全面的世界知识。为实现这一目标，我们在一个广泛的、高质量的和多样的语料库上训练模型。为了确保语言丰富性、主题多样性和跨语言及写作风格的覆盖，预训练数据集经过精心策划和过滤。模型在 10 万亿个标记上进行训练，使其能够对语言结构、事实知识和常识推理有深入理解，这为后续的微调和特定领域的适应奠定了坚实的基础。

与其前身相比，TeleBase2 在预训练期间使用了更广泛和更高质量的训练数据集。我们通过聚合来自多个来源的多样化、高质量数据来构建预训练语料库，以创建坚实的知识基础。数据集包括通用域和特定领域的内容。通用域来源包括网页、书籍、百科全书、新闻文章、社交媒体平台、学术论文、代码库和其他资源。特定领域的数据则从超过二十个专业行业中挑选出来，如金融、建筑、医疗保健和其他技术领域。

2.1.1 数据清洗

数据清理对于通过确保训练数据的质量、一致性和相关性来提高模型性能至关重要。我们实施了一套数据清理和质量保证技术，具体如下：

去重。我们实施了一种分层去重策略，包括 URL 级别、文档级别和段落级别的去重。通过借鉴类似 TeleChat (Wang et al., 2024b) 的方法，这种多层次框架确保在去除冗余数据的同时保持训练语料库的多样性和质量。

启发式过滤。我们设计了几种启发式方法来提升数据的整体质量。以下是一些启发式规则的列表。(1) 我们排除过于简短或缺乏实质性信息内容的文本。(2) 我们过滤掉标点符号频率异常高或低的文本。(3) 包含过多脏话的文本从数据集中排除。(4) 与代码相关的数据使用特定于源网站的评价标准进行处理。例如，GitHub 项目中的代码如果其仓库星数较低，则从数据集中排除。

基于模型的质量过滤。我们将大型语言模型（LLMs）整合到数据过滤流程中，以加强质量控制。在初步的自动化过滤步骤之后，我们部署 LLMs 进行深入的语义分析。这些模型评估文本的相关性、连贯性和流畅性，同时识别并标记可能具有攻击性、偏见或不当的内容。此外，它们还可以检测出微妙的问题，例如逻辑不一致、离题段落和可能未被基于规则的系统捕捉到的不自然语言模式。

数学和代码数据清理。对于数学和代码相关的数据，我们在质量保证中优先考虑正确性和可执行性。为了确保语法正确性，我们使用自动化脚本和静态分析工具来过滤出错误的数据。代码样例通过代码执行反馈进行验证，而数学问题则通过符号计算工具来确认方程式和解的准确性。我们还将大型语言模型（LLM）整合到我们的验证工作流程中。具体而言，LLM 负责审查代码逻辑、识别潜在错误，并生成预期输出以与原始数据进行比较。对于数学内容，模型独立解决问题并将其结果与提供的解进行交叉参考。这种混合方法能够有效发现传统基于规则的方法可能忽略的微妙错误。最后，一部分数据由人工专家进行人工审核，以确保其清晰、完整和相关性。这包括验证代码样例是否是自包含的并有良好的文档说明，而数学内容则遵循标准符号和格式约定。

2.1.2 确定数据组成

数据组成对模型性能有显著影响。然而，对于像 TeleBase2-115B 这样非常大的模型，进行广泛的数据组成调整是不可行的。为了解决这个问题，我们在较小的模型（3B 和 7B）上进行了一系列实验，以评估数据组合对模型性能的影响。例如，我们测试了训练中中英文语料库的不同比例，观察到英文语料的比例不应被过度减少。这一发现可以归因于两个因素：(1) 中文本身具有更高的语言复杂性，以及 (2) 中文语料库的总体质量低于英文。基于这些见解，我们预测了在不同数据组合下大模型的性能，并选择最有前景的组合进行训练扩展。

在模型训练过程中，我们采用课程学习策略来动态调整不同数据类型的比例。在初始训练阶段，我们强调使用更简单和更通用的数据，帮助模型建立语言理解和基本推理的坚实基础。随着训练的进行，我们逐渐增加更复杂和专业化的数据的比例，比如数学问题和与代码相关的任务，使模型能够逐步建立高级能力。为了确保学习的平衡性，我们每 1000 亿个 token 进行一次全面评估，使用涵盖所有主要数据类型的多样化基准测试或验证集。根据评估结果，我们在后续训练阶段调整数据采样比例，增加模型表现相对薄弱的数据类型的表示。这一动态调整过程使模型能够在所有领域保持稳定的改进，最终形成一个更强大和多功能的语言模型。

2.1.3 训练细节

我们采用 Adam 优化器来对 TeleBase2 进行预训练，并使用以下超参数设置：$\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 1 \times 10^{-8}$ 。使用余弦学习率调度器来调节学习率，根据模型大小缩放峰值学习率。在经过预热步骤达到其最大值后，学习率逐渐衰减至峰值的 10 % 。对所有模型参数（偏置项除外）应用权重衰减，系数为 0.01 。梯度裁剪以 1.0 的范数执行。所有可学习参数使用标准差为 0.006 的正态分布初始化。更多的超参数配置详见表 2 。按照 TeleChat1 的方法，我们将来自同一来源的数据进行串联处理，而不应用跨样本注意力屏蔽。在第一阶段的预训练中，我们将最大序列长度设置为 8K，并在 10T 个标记上对 TeleBase2 进行预训练。

为了优化训练效率与效果之间的平衡，我们在退火阶段整合了长上下文扩展。这种方法引入了一个统一的训练阶段，旨在同时提升基础模型的通用能力和长上下文理解能力。具体来说，我们将 TeleBase2-35B 的上下文窗口扩展到 256K 个标记，将 TeleBase2-115B 的上

HyperParams	TeleBase2-35B	TeleBase2-115B
Peak lr	3×10^{-4}	2×10^{-4}
Batch tokens	8M	6M
Warm-up fraction	0.001	0.001
Rms Norm Epsilon	1×10^{-5}	1×10^{-5}

Table 2: TeleBase2-35B 和 TeleBase2-115B 预训练阶段的超参数详细信息。

下文窗口扩展到 128K 个标记，同时保持它们与 8K 标记版本相同的通用能力。这种整合确保模型在适应扩展上下文需求的同时保持强大的基础技能。

2.1.4 数据管理

训练数据被分为五个不同的长度区间：0–8K，8K–16K，16K–32K，32K–128K 和 128K+。在每个区间内，数据按照领域进一步细分（例如，考试、网页、代码和其他类别），以便进行细粒度分析。在退火阶段，0–8K 区间以 7:3 的比例与其他区间组合，优先处理较短的序列，同时逐渐引入较长的上下文。同时，来自重要领域（如考试和代码）的高质量数据在所有长度区间中被上采样，确保对关键知识来源的全面覆盖。这种结构化的方法符合用于长上下文训练的数据工程原则 (Fu et al., 2025) 。

2.1.5 训练细节

预训练模型的上下文长度在不同阶段被顺序扩展，其中学习率根据余弦退火逐渐减少。第一个退火阶段的初始学习率与在 8K 预训练中使用的学习率相同。随后退火的进展基于先前训练阶段 1/3 步数的权重，并将当时的学习率作为初始值。在旋转位置编码（RoPE）中，基数是决定 LLM 有效上下文长度的关键因素之一 (Liu et al., 2024b) ; (Xu et al., 2024) ，因此我们将 RoPE 的基数设为 32K 退火时为 1×10^6 ，128K 时为 8×10^6 ，256K 时为 4×10^7 。此外，50B 的训练标记足以满足每个完整退火阶段的需要。经过多阶段的上下文扩展退火和微调后，TeleBase2 模型家族在 4K 到 128K 上下文长度的“捞针”测试（NIAH）中表现良好。图 2 展示了 TeleBase2-115B 的评估结果。

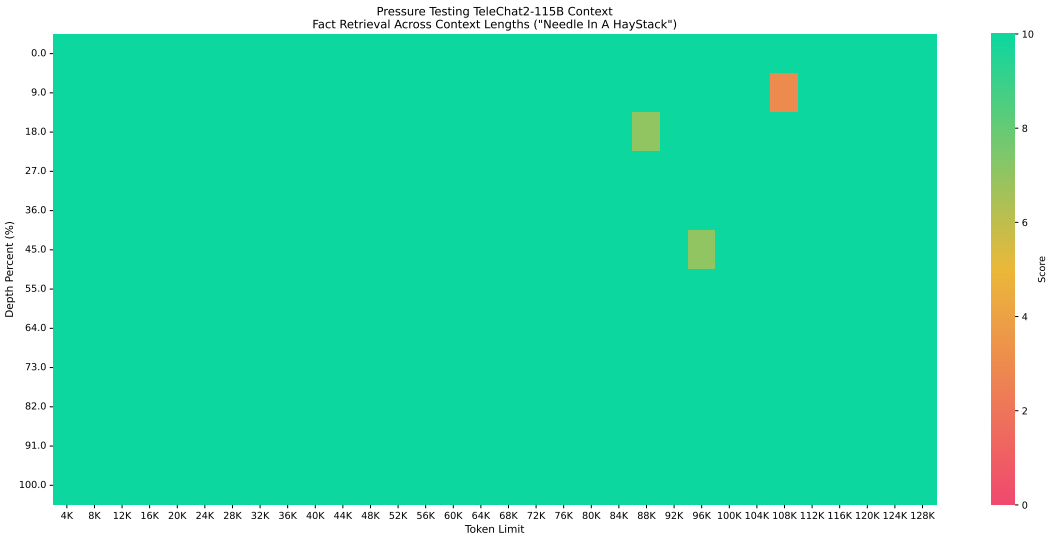

Figure 2: TeleBase2-115B 在“Needle In A Haystack”测试中的评估结果。

为了增强最终模型的鲁棒性和泛化能力，我们在训练过程后应用检查点平均化。具体而言，我们计算最后五个检查点的参数的逐元素平均值。通过平均这些检查点，我们有效地平滑了参数分布并提高了模型的稳定性。

对于我们的分词器，我们在 SentencePiece 中实现了带有字节级回退的 BPE，按照 (Touvron et al., 2023) 描述的方法将数字拆分为单个数字。我们通过加入特殊标记来区分对话角色并支持工具功能，从而扩展了最终的词汇表。为了在训练过程中确保计算效率并为将来可能需要的任何额外特殊标记预留空间，我们将模型的词汇表大小配置为 131072 。我们在所有 TeleChat2、TeleChat2.5 和 T1 模型系列中建立了统一的词汇表，以增强一致性和减少潜在的兼容性问题。

如图 3 所示，TeleChat2 模型通过监督微调（SFT）阶段和直接偏好优化（DPO）阶段直接在基础模型上进行训练，以增强其通用能力。另一方面，TeleChat2.5 和 T1 模型首先经历一个持续预训练阶段，然后进行三个阶段的后期训练过程。该过程包括：（1）包含思考和非思考模式的 SFT 阶段，（2）DPO 阶段以提高通用能力，以及（3）强化学习（RL）阶段以增强数学和编码能力。此管道产生 T1（思考变体）和 TeleChat2.5（非思考变体）。

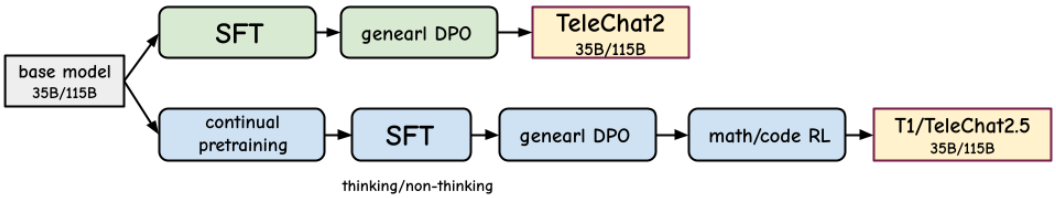

Figure 3: TeleChat2、TeleChat2.5 和 T1 的开发流程。

2.2 有监督微调

我们使用高质量、领域多样的数据（包括数学、编码、推理、对话、模型识别、安全性等）对预训练模型进行微调。为了创建高质量的 SFT 数据，我们开发了一个由两个阶段组成的流水线：（1）多样化的查询收集和（2）带有质量验证的响应生成。

2.2.1 查询收集

为了系统地组织和分类我们的 SFT 数据，我们开发了一个标记系统，通过领域和学科对提示进行分类，确保在各个不同主题领域的均衡表示。这个分层系统包括主要类别，如数学、编码、推理、对话安全、遵循指令、工具使用等。每个类别进一步细分为更细的分类，以全面捕捉所需的能力。

从公共数据集中获取数据。我们从广泛的开源数据集中获取查询，并采用严格的数据清理流程以消除重复或高度相似的条目。为了识别语义关系，我们将查询映射到高维嵌入空间，并应用 K-means 聚类算法有效地对其进行分组。

增强数据集的多样性和平衡性。在我们的标记系统中对数据进行清理和组织后，我们识别出某些类别的缺口，并观察到任务难度分布不均。为解决这些问题，我们利用自指令和指令进化技术来生成合成查询。这些方法使我们能够构建一个查询集，不仅全面涵盖知识系统，还能实现复杂性和多样性的良好平衡。具体而言，我们为不同的数据类别设计单独的难度评分提示，并利用 LLM 分别对每个来源内的每种数据类型进行评分。对于数学和代码等领域，我们采用通过率指标来区分学习难度。对于某些数据类型（例如创意写作、角色扮演、指令跟随和结构化数据生成），我们发现开源数据集中的难度水平普遍较低。为了解决这一问题，我们手动策划高质量的种子例子，并通过指令进化重建数据集。这种方法确保数据难度与现实世界使用复杂性紧密一致。

查询质量评分。为了保证查询质量，我们实施了基于大型语言模型（LLM）的评分机制。查询会根据预定义的标准进行评估，包括流畅性、标准化格式以及生成稳健响应所需的上下文信息的完整性。得分低的查询在训练中要么被排除，要么被降低权重，以优先考虑高质量的数据。在经过彻底审查和优化后，我们编制了一个具有广泛覆盖范围和良好校准难度范围的数据集。

2.2.2 响应生成与质量控制

我们采用人工标注和合成数据生成来生成响应。

人工标注收集。我们组建了一支由内部标注员和外部承包商组成的团队来进行人工数据标注。我们的标注人员在广泛的学科中具备多样的专业知识。为了应对当前大型语言模型（LLMs）面临挑战的查询，特别是在数学和推理任务中，我们依靠我们的标注团队生成高质量的回答。对于非推理任务，如创意写作、角色扮演和开放式问答，我们聘请人工标注员来验证合成数据的准确性。

合成数据生成。对于收集到的查询，我们首先使用高性能模型生成响应，然后根据特定任务的评估标准选择最佳答案。具体来说，对于可以验证正确性的任务（例如，数学、代码生成、指令执行、STEM 考试），我们采用基于规则的奖励系统，通过预定义的指标来评估响应，并且仅保留正确答案。对于主观任务（例如，人文学科、创意写作、开放式问答），我们利用 LLM-as-judge 框架，其中独立的大型语言模型根据流利度、一致性和相关性对响应评分。只有得分最高的响应会被保留。

我们实施了一整套基于规则的数据验证机制，以进一步确保数据的准确性。主要规则列举如下：（1）在生成过程中，常常会出现如重复内容、输出被截断以及字符不可辨认的问题。我们严格过滤掉此类错误数据。（2）通过基于规则的验证脚本，我们强制执行约束合规性，以确保符合诸如输出长度、段落数或用户查询提出的结构性指南等格式特定的要求。（3）我们使用敏感关键词数据库来实施内容过滤，以筛选出可能包含安全风险的答案。被标记的数据会进一步由人工注释者进行验证，以保证质量。

2.2.3 确定数据组合

后训练数据的组成对语言模型的表现具有关键影响。为了优化性能，我们采用了一种迭代算法，在最终数据组合中对高质量数据源进行上采样，同时对低质量数据进行下采样。我们的分析显示，模型性能与验证集 $\mathcal{V}$ 的困惑度之间可能存在负相关关系，该验证集是通过从我们的训练集 $\mathcal{T}$ 中提取 1% 的数据生成的。具体来说，取得更好评估性能的模型通常在验证集上表现出较低的困惑度。然而，当验证集按类别划分时，并不是所有子集在相同的训练步骤中达到其最小困惑度。为了解决这个问题，我们设计了一种算法，可以迭代地调整整个训练数据中每个类别子集 i 的表示比例 r_i ，其中 $i \in \mathbb{N}^*, 1 \leq i \leq |\mathcal{V}|$ 。

在第 t 轮微调实验中，我们根据分类将训练数据分成不同的子集，并在定期的训练间隔记录每个子集的困惑度。我们设置最大迭代次数为 T 以确保终止，$t \in \mathbb{N}, 0 \leq t < T$ 。接下来，我们使用三次样条插值来拟合一条曲线 $p = f_i^{(t)}(s)$ ，该曲线表示子集 i 的困惑度 p 随迭代 t 中训练步骤 s 的变化。将该曲线的最低点记作 $(s_i^{(t)}, p_i^{(t)})$ 。同样，我们根据每个子集的标记计算困惑度的加权平均值，并拟合一条最低点记作 $(\bar{s}^{(t)}, \bar{p}^{(t)})$ 的曲线。

新的比例可以按以下方式计算。

$$r_i^{(t+1)} = r_i^{(t)} \kappa^{\frac{s_i^t - \bar{s}^{(t)}}{\mu}}, \tag{1}$$

其中，κ 和 μ 是根据数据集特性动态校准的超参数，在我们的实验中，它们的最优值分别为 10 和 15,000。

$$\hat{r}_i^{(t+1)} = \frac{r_i^{(t+1)}}{\sum_{i=1}^{|\mathcal{V}|} r_i^{(t+1)}}. \tag{2}$$

正则化后，我们在下一轮实验中应用 $\hat{r}_i^{(t+1)}$ 作为新的比例。监督微调数据的数据分布如图 4 所示。

2.2.4 训练细节

我们通过网格搜索优化超参数以实现特定模型的训练配置。对于 35B 变体，余弦衰减学习率调度从 3×10^{-5} 开始，逐渐衰减到 1×10^{-5} ，批量大小为 8；对于 115B 变体，学习率从 1.5×10^{-5} 开始衰减到 1.5×10^{-6} ，批量大小为 16。为了提高训练效率和减少序列填充开销，我们实现了一种打包策略，将多个训练样本连接到一个序列中，同时在可能的情况下，将单轮样本战略性地组合成多轮对话，从而增强模型的多轮对话能力。

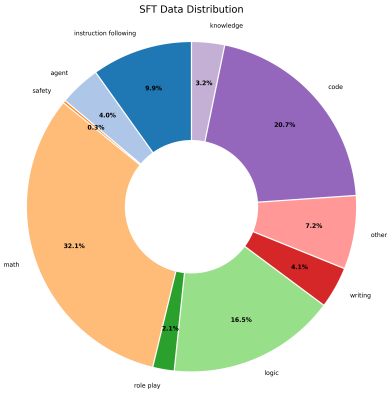

Figure 4: 监督微调数据的数据分布。

2.3 直接偏好优化

直接偏好优化（DPO, (Rafailov et al., 2023) ）是一种离线训练算法，旨在从偏好反馈中学习。它可以利用偏好数据直接优化策略，消除了构建奖励模型或在线从活动策略中采样的需要。DPO 的主要目标是最大化所选响应的对数似然和拒绝响应的对数似然之间的差距，同时确保模型保持接近初始策略 (Ivison et al., 2024) 。我们通过利用接受和拒绝输出的对来应用 DPO，使模型在一般任务上与人类偏好对齐，以抑制不良行为。在这一部分中，我们将在章节 2.3.1 详细说明我们的偏好数据构建流程，以及在章节 2.3.2 详细说明我们的训练细节。

2.3.1 偏好数据整理

偏好数据在提高大型语言模型的生成质量和性能方面至关重要。一个偏好数据集 P 通常包含提示、回复和排名。每个提示 x 与一对回复 y^+ 、y^- 相关联（其中 y^+ 是选择的回复，y^- 是被拒绝的回复），以及它们之间的偏好排名（表示为 $y^+ \succ y^- | x$ ）。我们生成偏好数据的过程包括四个阶段：提示选择、从模型池中生成回复、使用 LLM-as-a-judge 注释偏好，以及构建对。此过程的具体细节如下所示。

提示选择。为直接偏好优化（DPO）准备数据集的第一步是选择用于生成响应和收集偏好的提示或用户指令。提示集的质量和多样性对于确保 DPO 的有效性至关重要。具体来说，一个高质量的提示集应满足两个关键标准：（1）它应展示多样性并涵盖广泛的领域，包括数学 & 推理、编码、创意写作等，以增强模型的适应性，使其能够应对各种实际挑战和问题。(2) 它应包含各种简单、中等和挑战性的问题，以促进全面的理解并减少在特定难度水平上过拟合的风险。

我们将 SFT 提示分为两部分，分配 90 % 用于 SFT 训练，10 % 用于 DPO。由于我们的 SFT 提示提供了对多样化领域和不同复杂度级别的全面接触，因此我们的 DPO 提示能够满足上述标准。另外，我们整合了新的指令遵循约束，以增强模型遵循指令的能力，并基于之前模型的不足之处引入了新的成对提示来应对其缺点。

响应生成。当给定一个提示时，我们首先从大量最先进的开源和专有模型中进行抽样，这些模型在参数大小和模型家族上有所不同。我们使用贪婪抽样法，并且对每个模型仅抽样一次。接下来，我们通过从最新的 TeleChat2.5 和 T1 模型中抽样补全，结合在线策略数据，并利用高温采样来生成多种响应。为了提高拒绝采样的效率，我们使用 vllm (Kwon et al., 2023) 来加速推理过程。

偏好注释。在为每个提示生成多个回答后，必须为每个回答分配一个奖励。(1) 对于可验证的问题，奖励是基于特定标准或规则来确定的。例如，在编码问题中，我们评估解决方案是否通过单元测试。在数学、推理和标准考试问题中，我们评估生成的答案是否导致正确的解决方案。对于需要遵循说明的受限提示，我们核实生成的答案是否符合限制条件。(2) 对于具有自由形式答案的开放性问题，我们使用一个大型语言模型作为评判者 (Zheng et al., 2023) ，根据四个不同的因素，对每个答案进行 0 到 10 的评分，这些因素是: 有用性、遵循说明程度、完整性和准确性。

偏好对的构建。偏好对的构建遵循几个关键原则。

- 所选择的回答仅从得分最高的回答中挑选。为了维持回答质量标准，我们规定当选回答的得分必须不低于 ≥ 8。当多个回答获得相同的最高分时，优先考虑由 TeleChat 系列本身生成的回答，而非其他策略产生的候选项。这个设计选择减少了 DPO 训练中固有的潜在分布偏移问题，如之前工作 (Rafailov et al., 2023) 所示。
- 被拒绝的响应严格取样自 TeleChat 系列模型自身生成的结果。这种方法允许模型通过学习自身的错误模式来自我修正。

在选择和拒绝的对之间实施一个最小绝对分数差（$\Delta \geq 2$ ）。这个阈值考虑到 LLM 作为评判者评分的不稳定性，有效地过滤掉那些由于分数微小变化而可能不反映真实质量差异的模糊比较。对于生成多个有效对的输入提示，我们会随机抽样每个输入提示中的 $K = 4$ 个不同对。这为 DPO 训练产生了 $98,273$ 个对。

2.3.2 训练细节

我们针对 DPO 训练了一个 epoch，学习率为 5×10^{-7} ，批量大小为 256 。我们使用学习率预热和余弦学习率调度器。β 超参数设置为 0.1 。我们在长上下文 SFT 检查点上进行 DPO 训练，但只选择令牌长度短于 8,192 的样本。我们的观察表明，在 DPO 中仅使用短上下文的训练数据不会对长上下文性能产生负面影响。

在 DPO 训练期间，我们为对的赢家添加了一个额外的负对数似然（NLL）损失项，缩放系数为 0.2，这对性能也证明至关重要 (Pang et al., 2024) 。另外，我们在损失函数中使用了一种技术，将选中和拒绝的响应中的终止标记屏蔽，以增强 DPO 训练的稳定性，类似方法如 (Grattafiori et al., 2024) 中描述。这是必要的，因为在选中和拒绝响应中存在共享标记，产生了一个对立的学习目标，要求模型同时增加和减少这些标记的可能性。

2.3.3 模型合并

我们在 DPO 阶段将涉及不同数据版本或超参数的实验中派生出的模型合并。特别是，我们通过简单地平均权重来合并多个模型，观察到这一合并过程有利于增强模型的鲁棒性和整体能力。

2.3.4 迭代 DPO

离线偏好调优方法程序的迭代应用已被证明是有利的，更新的模型用于构建更加具有信息量的新偏好对，带来进一步的改进。因此，我们在三个周期中应用这些方法，在每个周期中通过从最新模型中采样合成数据来收集新的偏好对。

2.4 强化学习

强化学习（RL）已被证明在超越监督微调（SFT）阶段后，提高大型语言模型（LLMs）的推理能力方面是有效的。(Shao et al., 2024) 在这项工作中，我们专注于通过强化学习策略优化模型在数学推理和代码生成中的表现。

(1) 数学 RL。我们从两个公开可用的来源整理了一个数据集：OpenR1-Math-220k [2] 和 Synthetic-1 [3] 。为了确保数据质量，我们过滤掉需要证明的问题以及那些引用不完整或不一致的问题。具体来说，我们只保留可以使用 math_equal 函数 [4] 自动验证的问题，该函数用于检查答案的数值或分析等价性。对于答案提取，我们提示模型将最终答案用 boxed { } 包裹，并运行验证过程以确认正确性。

(2) 编码 RL。我们从 SFT 数据集中提取了一部分编码问题，只保留能够执行代码执行反馈的样本。对于单元测试，我们开发了一个安全的本地代码沙箱环境，支持多种测试方法，包括标准输入输出验证和基于断言的验证。

(3) 工具使用 RL。我们遵循两步策略策划强化学习的函数调用数据：（i）初始候选集构建。我们选择一批源自与监督微调（SFT）数据相同来源的函数调用数据作为候选数据。随后，

[2] https://huggingface.co/datasets/open-r1/OpenR1-Math-220k
[3] https://huggingface.co/datasets/PrimeIntellect/verifiable-math-problems
[4] 可在 https://github.com/hkust-nlp/simpleRL-reason/tree/v0 获得

使用多个大型语言模型（LLMs）对每个查询进行多次推理。选择那些在不同模型中输出一致的查询及其对应的真实答案作为训练输入。(ii) 难度分层和数据策划。使用目标模型对查询进行多次推理。将模型输出与参考答案进行比较，以计算 pass@5 率。根据 pass@5 将查询分为不同的难度级别：

- pass@5 = 1 ： 这些查询对当前模型来说太简单了（简单）。
- 0 < pass@5 < 1 ： 该模型有潜力正确回答，但在这些查询上表现不稳定（中等）。
- pass@5 = 0 ： 这些查询对于模型来说很难正确回答（困难）。

强化学习训练数据集由中等和困难数据组成，比例为 2:1。对于奖励函数设计，我们基于数据类型实施类别特定处理。具体来说，数据被划分为需要工具和无需工具的类别，计算公式如下：

$$\text{reward} = \begin{cases} 1, & \text{if } I_{\text{tool}} = 1 \wedge M_{\text{format}} = 1 \wedge M_{\text{match}} = 1 \\ -1, & \text{if } I_{\text{tool}} = 1 \wedge (M_{\text{format}} = 0 \vee M_{\text{match}} = 0) \\ 2 \times \dfrac{S - S_{\min}}{S_{\max} - S_{\min}} - 1, & \text{if } I_{\text{tool}} = 0 \end{cases}$$

我们的奖励函数设计基于任务是否需要调用工具进行区分。对于需要调用工具的任务，我们设定了一个二元的奖励机制：如果模型的输出格式完全正确，并且工具调用的具体内容与参考答案完全匹配，则给予全部奖励（+1 ）；如果输出格式不正确或者工具调用内容偏离参考答案，则给予惩罚（−1 ）。对于纯文本任务，不需要调用工具的情况，我们采用相对灵活的评分机制：首先，我们使用另一个大型语言模型（LLM）对模型的输出进行质量评估，得到一个原始质量分数 S ；然后，我们通过一个线性转换公式将这个原始分数映射到统一的奖励值范围 $[-1, 1]$ ，以便与工具调用任务的奖励进行统一比较和优化。

我们利用 OpenRLHF [5] 框架进行训练，并采用增强 ++ 算法 (Hu et al., 2025) 。为了确保稳定的训练，我们实施了动态采样，该方法会持续采样，直到批次完全由其准确率既不是 0 也不是 1 的样本填充，如 (Yu et al., 2025) 中所提出的。对于超参数，我们使用 AdamW 优化器 (Loshchilov & Hutter, 2019) ，学习率设为常数 5×10^{-7} ，并结合线性预热，持续 20 个回合步。在回合阶段，提示的批次大小设置为 128，每个提示生成 16 个响应。对于训练，微批次大小也配置为 128。

我们特别努力提高特定能力的性能，包括代码、推理、工具使用、长上下文和精确指令的遵循。

2.5 代码

两阶段训练策略。我们实现了一种由粗到细的两阶段微调方法。在第一阶段中，基础模型在数千万个从大型开源数据集（例如，CodeAlpaca、CodeSearchNet）和从 GitHub 代码库中提取的代码合成的多样化指令样本上进行训练。这个基础阶段通过让模型接触到广泛的任务范围来拓宽其能力。在随后的微调阶段，我们使用高质量、精心策划的指令数据集。这些包括多语言代码生成任务、编程竞赛（通过网络爬虫从 Codeforces 和 LeetCode 获取）和编程教程。对于每个查询，LLM 生成多个候选响应。可验证的问题通过代码执行反馈进行评估，而不可验证的问题则利用 LLM 自身来对候选响应进行排名并选择最合适的示例进行监督微调。

代码执行反馈。对于支持测试用例验证的问题，我们使用大型语言模型自动生成 10 个测试用例。这些测试用例全面覆盖正常场景、边界条件、异常情况和复杂输入，以严格评估正确性。测试用例按编程语言分类（例如，Python、C、C++、Java、JavaScript）并在安全的沙箱环境中执行。代码正确性通过运行时执行验证进行验证。由于代码执行中的错误（例如，无效语法或断言错误）导致失败的样本会被过滤掉，以确保训练数据的质量。

课程学习。我们实现了一种模型驱动的课程学习策略，该策略利用模型自身的生成能力在第二阶段训练时评估提示难度。具体而言，我们使用高采样温度（例如，$T = 0.6$ ）为每个提示生成十个响应。通过代码执行反馈确定的通过率被计算为难度的代理，这动态构建了一个训练课程。起初，模型专注于通过率较高的提示，确保稳定学习和基础技能获得。随着训练的进行，它逐渐过渡到通过率较低的提示，迭代地改进其编码能力，同时系统地扩展其边界。

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

2.6 数学与推理

双阶段训练策略。在数学和推理任务中，我们采用与代码任务一致的双阶段微调策略，从广泛的能力构建过渡到深入的精确优化。在第一个阶段，基础模型在超过一千万个合成样本上进行训练，这些样本来源于广泛的开源数据集（例如，StackExchange）、带有答案的合成 K-12 数学问题以及合成的大学教学材料。所有数据都经过来源质量评估、去重、格式清理、合成数据生成和质量抽样检查。第二阶段使用更小但质量更高的精选数据集。逻辑推理样本是手动收集的，带有真实答案，涵盖因果推断、运筹学和博弈论等领域。数学数据包括高质量的开源数据集（例如，MATH，GSM8K 训练集）、带验证答案的授权 K-12 数学问题、全球竞赛问题（例如，IMO，AMC），以及少量合成数据以平衡分布。所有样本均经过三重验证：问题质量评分、答案一致性检查和推理过程验证。一个难度分级机制确保不同难度级别的数据分布平衡。

答案验证机制。为了验证数学答案的准确性，我们实施了一种多模型协同验证策略，并结合人工监督进行共识筛选。具体来说，对于一组目标数学问题，我们使用多个大型模型独立生成答案。专门的答案一致性判断机制会对输出进行分析和比较。对于所有模型输出完全一致的样本，将进行人工抽样质量检查，而不一致的输出则通过人工标注重新检查，以确保最终答案的正确性。

2.7 工具使用

数据管理。我们收集主流开源函数调用数据集 (Zhang et al., 2024a) ("interstellarninja") (Qin et al., 2023) (Toshniwal et al., 2024) (Li et al., 2023a) 并进行数据清理和重构。我们的验证主要关注两个关键方面：

- 格式验证，即我们严格检查工具调用与提供的功能列表的对齐情况。这包括验证：1）工具名称的正确对应，2）参数名称的匹配，以及 3）参数类型符合要求。
- 工具调用结果验证，我们利用大型语言模型（LLM）来评估工具调用的有效性以及工具名称和参数配置的准确性。

此外，参考构建 BFCL 基准所使用的方法，我们对收集的函数调用数据进行分类，以确保训练数据集中函数调用类型的均衡分布。

基于工具图的数据构建。清洗开源数据后，我们收集了约 11 万个样本。然而，在清洗过程中，我们发现问题包括中文数据不足、对话轮次有限和难度水平低。为了解决这些问题，我们基于 API 之间的依赖关系构建了一个工具图结构，利用各种图采样方法创建具有平衡难度分布的任务。此外，我们利用工具图中的依赖关系来辅助验证多轮工具调用的准确性，显示出显著的优化效果。

为了提高模型的指令跟随能力，我们开发了一个系统化的流程来构建 SFT 训练数据集。在此过程中，我们通过三个关键阶段构建高质量的训练数据：约束集构建、指令进化和带验证过滤的响应生成。

约束集构建。按照 IFEval (Zhou et al. (2023)) 的方法，我们识别出具有代表性的应用场景，并构建一个完全由可验证约束组成的约束集，这些约束可以通过自动脚本进行严格验证。例如，这些约束包括响应长度要求、语言规范、格式指南等。通过利用自动化验证，这种方法消除了手动干预的需求。

指令进化。基于约束集合，我们通过显式地结合随机抽取的一组约束（通常不超过六个）来提示 LLM 将种子指令进化为新指令。这些约束指导 LLM 生成具有明确操作要求的指令。此外，要求 LLM 明确指定与这些约束对应的参数值（例如，关键词数量，字数限制），这些参数值被记录下来以便后续验证。

通过验证过滤生成响应。最后，我们利用 LLM 为新构建的指令生成响应。结合每个指令相关的约束定义和参数值，我们为每种约束类型设计了专门的验证脚本。这些脚本基于执行反馈评估模型的输出，并自动过滤掉不符合约束的响应。此过程确保生成的指令-响应对始终符合预定义的质量标准。

我们描述了支撑大规模预训练的硬件和基础设施，并介绍了几项优化，这些优化提高了训练效率。

之前版本的 TeleChat

2.8 基础设施

是在拥有 640 个 NVIDIA A100 GPU 的集群上进行训练的。随着我们扩展到新的 TeleChat 系列，训练迁移到了天翼云的上海计算中心，该中心提供了训练万亿级模型所必需的计算能力。

计算。新的 TeleChat 系列在最多包含 8k 张 Ascend NPU 的 Atlas 900 A2 集群上进行训练。集群中的每个节点包含 8 个通过 HCCS 连接的 NPU。训练任务是通过一个基于 MindCluster 的平台进行调度的。

存储。存储资源包括集群管理（CM）节点、元数据服务器（MDS）节点、对象存储服务器（OSS）节点以及被称为 OceanDisk 的物理存储设备。CM 节点通过双 25 Gbps 链路连接到基于云的存储系统，为分布式存储操作提供管理接口。OceanDisk 设备通过光纤通道（FC）网络直接连接至 MDS 和 OSS 节点，确保数据存储和检索的高速低延迟通信。这四种类型的节点和设备共同组成了高性能文件系统（HPFS）共享存储系统，该系统针对分布式和高吞吐量的工作负载进行了优化。HPFS 共享存储系统通过双 100GE 链路上行连接到聚合以太网上的 RDMA（RoCE）交换机，能够无缝集成到更大的网络基础设施中，并确保计算和存储节点的高带宽访问。

网络。参数通信网络采用两层 Clos 结构（Charles Clos 拓扑）。每个训练服务器将其 200GE 上行链路连接到 RoCE 交换机，实现处理单元之间高速 200GE RoCE 互连。Spine/Leaf 层级使用非收敛设计配置，以确保最大带宽可用性。参数通信网络整合了网络侧负载平衡（NSLB），以在大型模型训练期间确保链路层的有效负载平衡。这种方法减少了哈希冲突并提高了计算集群的总体吞吐效率。

Telechat2 的分布式训练基于 MindSpore 通用大型模型并行框架 (MindSpore, 2025) 提供的 4D 并行策略。该框架通过集成四种关键的并行策略：数据并行（DP; Rajbhandari et al. (2020)；Zhao et al. (2023)）、张量并行（TP; Shoeybi et al. (2020)）、流水线并行（PP; Huang et al. (2019)；Narayanan et al. (2021)）和上下文并行（CP; Liu et al. (2023a)），旨在支持高效且可扩展的大规模模型训练。

数据并行 (DP): 输入数据集沿批处理维度进行划分，不同的设备组独立处理不同的数据批次。在反向传播过程中，所有设备之间进行梯度同步，确保模型参数的一致更新。该方法对于扩展到更大数据集以及提高分布式系统的硬件利用率特别有效。

张量并行（TP）：模型权重在设备之间进行分区，以减少内存使用和计算开销。使用诸如 All-Gather 和 ReduceScatter 之类的集体通信原语来交换和聚合中间结果，从而实现高效的张量操作分布式计算。

流水线并行（PP）：模型被划分为多个层或阶段，每个阶段分配给一个特定的设备组。前向和后向传递以流水线方式执行，以最大化并行性。为了缓解流水线气泡造成的低效，采用了负载均衡和虚拟流水线调度等策略。

上下文并行（CP）：这一策略是 MindSpore 独有的，实现了一种 3D 序列并行方案，旨在高效处理长序列任务。通过在设备之间分割序列计算，CP 缓解了与大输入序列相关的内存和计算限制。

为了确定分布式并行的最佳参数，我们在各种配置下进行了广泛的实验。张量并行（TP）由于诸如全收集（All-Gather）和规约散射（ReduceScatter）等操作会产生通信开销，而流水线并行（PP）则受泡（Bubble）和发送/接收（Send/Recv）通信带来的低效影响。通过采用负载平衡和其他优化技术来减少流水线泡，我们发现 PP 并行在效率方面始终优于 TP。在仔细调整了并行配置、硬件资源和软件优化后，我们在表 ?? 所示的配置中实现了 33.8 % 至 36.3 % 的模型浮点操作利用率（MFU; Chowdhery et al. (2022)）。

在大规模分布式训练中，精确控制全局批量大小对于确保模型收敛和实现最佳性能至关重要。众所周知，过大的批量大小会对收敛动态和最终模型质量产生不利影响。因此，在训练初期，全局批量大小通常限制在 4M 到 8M 令牌之间。当在 4096-NPU 群集上训练 Telechat-115B 时，数据并行 (DP) 维度的增加导致每批次的令牌增多。为了将每批次令牌限制在 4M，减少了管道中的微批次数量，这增加了管道气泡并降低了整体效率。为了解决这个问题，我们利用虚拟管道并行性 (VPP) 功能来最小化气泡，从而实现了 33.8 % 的 MFU。当扩展到 6144-NPU 群集时，我们将 VPP 因子增加到 3，进一步减少了管道气泡率，并将 MFU 提升至 34.1 % 。对于序列长度为 128k 的超长序列训练，我们利用上下文

并行 (CP) 来缓解与长序列相关的内存和计算压力。此方法使得在 4096-NPU 群集上训练 Telechat-115B，实现了 34.5 % 的 MFU。

这些结果展示了在优化分布式训练效率时，仔细平衡并行策略和利用高级特性（如 VPP 和 CP）的有效性，特别是在扩展到大型集群和处理长序列数据集时。

除了这些基础的并行策略之外，Telechat 的分布式训练还结合了由 MindSpore 启用的几项高级优化。选择性重计算用于通过在反向传播过程中重新计算选定的激活而不是存储它们来减少内存开销。优化器并行性通过将优化器操作的计算工作量分布到不同设备上来提高训练效率。细粒度多副本功能允许计算和通信的重叠，有效地掩盖了通信延迟并提高端到端的吞吐量。此外，流水线并行优化利用虚拟流水线并行性 (VPP)，采用 1F1B（一次前向，一次后向）调度策略，结合流水线负载平衡调整，以实现更高的计算资源利用率。

选择性重新计算。在大规模模型训练中，前向传播中生成的激活通常会被存储以供反向传播使用，这会导致显著的内存消耗。在流水线并行（PP）中，这个问题更加严重，因为来自多个微批次的激活必须被累积，进一步增加了内存压力。对于超过 70B 参数的模型，常用的方法是省略激活存储并在反向传播期间重新计算激活，从而减少内存使用。然而，这种方法在反向传播时引入了额外的计算，可能降低计算效率。

为了解决此问题，新一系列的 TeleChat 训练利用了 MindSpore 提供的选择性重计算能力。这种方法有选择性地对关键操作符进行重计算，平衡了内存节省与计算开销。具体来说，我们针对前馈网络（FFN）中的操作符，包括 Silu 和 Mul，以及 RMSNorm（均方根归一化）中 Cast 操作符（从 fp32 到 bf16）。这些操作符因其计算成本低且对减少激活分配的内存影响显著而被选中。该策略使我们在保持训练效率的同时优化了内存使用。

此外，MindSpore 支持通信感知选择性重新计算，当与优化器并行结合时，可以实现类似于 Zero3 的效果。MindSpore 还支持分层重新计算、选择性重新计算和通信感知重新计算，并进一步与流水线并行优化整合。这些先进技术共同优化内存分配和计算，确保大型模型的高效训练。

优化器并行。在数据并行训练中，参数更新是在设备之间冗余计算的，这导致了内存使用效率低下并在大规模网络中表现欠佳。优化器并行通过在数据并行维度上分配优化器计算来解决这个问题。具体来说，模型参数和梯度是基于设备 ID 被划分为多个切片，每个设备独立更新其分配的切片。更新完成后，参数通过通信操作在设备之间进行聚合。这种方法具有自然负载均衡的好处，确保每个设备承担相等份额的参数和计算。然而，它对参数形状提出了必须能被设备数量整除的限制。这种方法的理论收益与参数分片相符，并且在 TeleChat 的分布式训练中引入了一些优化以提高其有效性。

- 用于静态内存减少的权重分片：模型权重被分割以进一步减少静态内存消耗。为了在前向和后向传播过程中保持原始张量形状，共享权重在每次迭代结束时进行聚合，并在下一次迭代的前向传播之前重新分配。
- 重叠通信以提高性能：优化器并行的一个主要缺点是与共享权重相关的通信开销。通过将通信操作与前向计算重叠，我们可以最小化感知到的通信延迟。具体来说，跨迭代执行通信允许通信操作符被分组和融合，从而实现通信和计算的高效交错。

流水线并行优化。在流水线并行训练情境中，内存不平衡是一个突出的挑战，特别是前端阶段经常面临显著的内存压力。为了解决这个问题，我们实施了一种优化策略，将调整每个阶段分配的层数与差异化的重计算策略相结合：

- 内存密集型阶段：对于经历高内存压力的阶段，我们减少了分配给这些阶段的层数并采用选择性重计算的方法来处理所有层。这种方法在最大限度节省内存的同时平衡了计算方面的权衡。
- 轻内存阶段：相反，内存压力较小的阶段被分配了额外的层，并仅对一部分层采用选择性重新计算，从而在内存使用和计算效率之间取得平衡。

为了确保大规模模型训练的有效性，批次 token 大小通常受到限制（例如，8M 或 16M）。在用大型集群训练时，数据并行性（DP）的显著增加导致微批量大小减小。在固定数量的管道阶段下，较小的微批次会导致更大的管道空泡，负面影响训练效率。为了提高管道并行性的效率并减少空泡比例，我们在 TeleChat2 模型的 115B 参数训练中采用了虚拟管道并行（VPP）。传统的管道并行通常将连续层（例如，Transformer 层）分配到一个阶段。相反，VPP 调度在每个阶段内交错计算非连续层（图 5 ）。通过增加通信开销，这种方法显著减少了空泡比例，从而改善了整体训练性能。

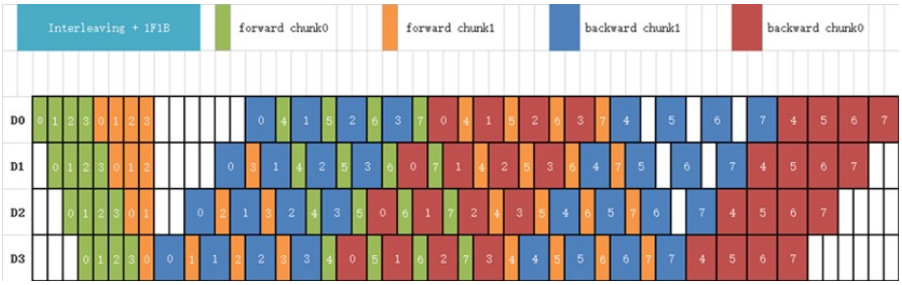

Figure 5: 虚拟流水线并行（VPP）调度示例：演示了使用 IF1B（一次前向一次后向）策略进行非连续层的交错计算。该调度机制通过重叠通信和计算阶段来减少流水线泡沫，同时保持各阶段的负载平衡。

Table 3: 预训练期间的硬件故障统计

Failure Category	Count	Proportion (%)
Optical Module Failure	36	19
HBM Failure	33	18
Memory Failure	14	8
DPU Failure	14	8
AI Core Failure	6	3
Optical Cable Failure	5	3
Motherboard Failure	5	3
Hard Drive Failure	3	2
Overheating	3	2
CPU Failure	2	1
NPU Failure	2	1
Power Supply Failure	2	1
RAID Failure	1	1
Controller Failure	1	1
Others	57	31

长序列优化。为了支持长度为 128k-256k 标记的长序列训练，我们通过分割查询、键和值（QKV）张量的序列维度实现了序列并行（也称为上下文并行）。这种方法有效地减少了内存消耗。在注意力计算阶段，使用全体聚集通信重新组合键和值张量的序列维度。

为了实现序列负载均衡，我们利用点对点的全收集通信来交换不同设备上的查询和注意力结果的序列维度数据。这使得可以将后期阶段计算密集的序列与前期阶段较轻的序列进行交换，从而确保设备间的计算负载均衡。

对于更长的序列（例如，数百万个标记），我们采用了 MindSpore 提供的 Ring-Attention 算法。该方法在 Attention 计算过程中，避免了对键和值张量的序列维度进行完全重组。相反，它对局部的 QKV 数据进行块计算，确保了数学等效，同时实现了完全负载均衡以及计算与通信的重叠。该优化进一步减少了内存消耗，并在训练超长序列时提升了性能。

在 TeleChat2 的预训练阶段，硬件故障是服务中断的主要原因，包括光学模块、HBM（高带宽内存）和内存组件的问题。为应对这些挑战，我们实施了以下措施：

恢复机制优化。通过改进日志、检查点和数据的存储和加载机制，同时升级训练框架和调度平台，实现了优化的故障恢复。这些改进显著减少了中断后恢复训练所需的时间，并通过版本检查预先解决了集群环境问题。

硬件可靠性改进。强化了对关键硬件（如 HBM、光学模块和内存）的检查程序。此外，制定了更严格的硬件更换标准，并简化了硬件问题解决流程。

由于这些努力，预训练中后期的每周故障率保持在低于 1%。由于硬件故障引起的训练中断显著减少，核心集群硬件的平均故障间隔时间（MTBF）为 4 天，最大间隔为 21 天。集群的可用性指标表现强劲，每周正常运行时间始终超过 99%，最长的连续训练时间长达 288 小时。

尽管取得了显著的改进，但仍有几个关键挑战尚未解决，这些问题继续阻碍着训练可靠性和效率的进一步提升。缺乏有效的故障诊断工具导致了一系列问题，例如解释错误代码的困难、定位错误根节点的挑战，以及在空间和性能方面无法有效监控共享存储利用率。这些缺陷不仅延长了发生故障时的停机时间，还增加了解决问题和系统恢复的复杂性。开发强有力的诊断工具和监控系统以解决这些空白，将对于最小化中断、优化资源利用并确保分布式训练环境中的无缝扩展至关重要。

3 评价

3.1 预训练模型

在一个多样的基准测试中评估 TeleBase2 的表现，这些基准测试是基于内部评估框架设置的。对于基础模型，评估重点放在它们在一般知识、常识、逻辑推理、数学问题解决和编程能力方面的表现。我们评估的基准测试列表如下：

- 常识基准包括 C-Eval (Huang et al., 2023) （零样本）、MMLU (Hendrycks et al., 2021a)（五样本）、MMLU-pro (Wang et al., 2024a)（五样本）、CMMLU (Li et al., 2023b)（五样本）、GAOKAO (Zhang et al., 2024b)（零样本）、AGIEval (Zhong et al., 2023)（零样本）、GPQA (Rein et al., 2023)（五样本）和 TheoremQA (Chen et al., 2023)（五样本）。
- 常识基准包括 CommonsenseQA (Talmor et al., 2019)（5-shot）和 TruthfulQA (Lin et al., 2022)（zero-shot）。
- 逻辑推理基准包括 BBH (Suzgun et al., 2022)（3 次示例）和 HellaSwag (Zellers et al., 2019)（零次示例）。
- 数学问题解决基准包括 GSM8K (Hendrycks et al., 2021b)（4-shot）、MATH (Hendrycks et al., 2021c)（4-shot）和 Ape210K (Zhao et al., 2020)（1-shot）。
- 编码基准测试包括 HumanEval (Chen et al., 2021)（零样本），MBPP (Austin et al., 2021)（三样本），Humaneval+（零样本），MBPP+（三样本）(Liu et al., 2023b)。

在表格 4 中，我们比较了在上下文长度 8K、32K 和 256K 处训练的 TeleBase2-35B 与 Qwen2.5-32B-base。在表格 5 中，我们比较了在上下文长度 8K、32K 和 128K 处训练的 TeleBase2-115B 与 Qwen2.5-72B-base。所有模型使用自定义评价框架在标准化设置下进行了系统地评估，以确保公平和严格的比较。

为了全面评估指令调优模型的质量，我们利用自动化评估框架来评估思考模型（T1）和非思考模型（TeleChat2 和 TeleChat2.5）的性能。这些指令模型根据以下基准进行评估，以比较它们的能力。

- AlignBench (Liu et al., 2024a) 是一个综合性、多维度的基准，用于评估中文大型语言模型（LLM）与人类价值观和现实世界需求的一致性。它包含 8 个核心类别，683 个真实场景查询和人工验证的参考。
- IFEval (Zhou et al., 2023) 是一个评估大型语言模型遵循可验证指令能力的基准。它提供了 25 种指令类型和大约 500 个提示，每个提示都有可量化的标准。
- BFCL（伯克利函数调用排行榜）(Patil et al., 2025) 是一个基准测试，旨在评估大型语言模型（LLM）的函数调用和工具使用能力。该基准测试采用多维度的评估方法，包括单轮函数调用、多轮函数调用和幻觉检测。本论文中呈现的 BFCL 基准测试结果特别反映了在 python-ast 轨道上的单轮性能，报告了实时和非实时子任务的平均值。
- MATH500 源自最初的 MATH 数据集 (Hendrycks et al., 2021c)，该数据集包含 5K 道数学题目。

对于 T1 模型，我们采用采样温度为 0.6，top-p 为 0.95，top-k 为 50，以及重复惩罚为 1.05。对于 TeleChat2 和 TeleChat2.5 模型，使用贪婪搜索和重复惩罚 1.01。对于两种模式，我们将最大输出长度设置为 32,768 个标记。TeleChat 模型系列的评估结果，以及在类似设置下与其他具有可比参数规模的模型的比较，详见表格 6 和 7 。

评估结果显示，TeleChat 系列模型在思考模式和非思考模式下均展现出强大的能力。T1-115B 在思考模式下表现出色，在 MATH500 测试中超过 OpenAI o1-mini 4.0 分（94.0 对

Table 4: 比较在 8K、32K 和 256K 上下文长度下训练的 TeleBase2-35B 与 Qwen2.5-32B 基础模型的性能。

Benchmark	TeleBase2-35B-8K	TeleBase2-35B-32K	TeleBase2-35B-256K	Qwen2.5-32B
General Knowledge				
C-Eval	87.2	87.8	86.2	86.1
MMLU	72.4	74.2	71.0	75.6
MMLU-pro	47.0	48.4	43.0	62.1
CMMLU	77.2	77.9	76.7	88.3
GAOKAO	68.6	63.2	59.1	52.1
AGIEval	68.9	71.3	69.3	82.7
GPQA	36.5	37.8	38.0	41.5
TheoremQA	41.0	42.8	40.3	44.3
Commonsense				
CommonsenseQA	88.4	85.7	85.3	83.4
TruthfulQA	57.2	54.0	55.0	70.0
Logical Reasoning				
BBH	81.7	82.6	82.5	70.0
HellaSwag	96.2	91.6	90.2	93.0
Mathematical Problem-Solving				
GSM8K	85.2	86.2	86.3	75.0
MATH	69.2	71.6	70.0	61.2
Ape210K	66.8	66.0	67.0	65.5
Coding				
HumanEval	73.8	70.7	73.8	78.0
MBPP	65.2	65.2	68.9	74.0
Humaneval+	66.0	66.0	67.4	69.5
MBPP+	70.5	71.4	70.5	70.5

Table 5: 在 8K、32K 和 128K 上下文长度下的 TeleBase2-115B 与 Qwen2.5-72B 基础模型之间的比较。

Benchmark	TeleBase2-115B-8K	TeleBase2-115B-32K	TeleBase2-115B-128K	Qwen2.5-72B
General Knowledge				
C-Eval	94.0	92.3	91.0	89.5
MMLU	81.0	79.9	78.9	77.2
MMLU-pro	53.2	53.0	52.5	63.8
CMMLU	82.0	81.3	80.0	90.3
GAOKAO	73.6	72.3	73.7	68.9
AGIEval	69.7	70.0	71.8	84.7
GPQA	41.3	41.3	38.3	40.3
TheoremQA	44.8	45.8	45.3	46.5
Commonsense				
CommonsenseQA	86.7	85.3	85.7	87.1
TruthfulQA	62.6	61.6	61.0	71.0
Logical Reasoning				
BBH	81.5	82.7	82.8	85.1
HellaSwag	97.4	92	92.6	96.8
Mathematical Problem-Solving				
GSM8K	90.3	84.5	86.0	76.5
MATH	72.0	74.0	72.4	62.0
Ape210K	68.8	72.0	67.7	66.5
Coding				
HumanEval	72.6	69.5	67.7	78.7
MBPP	70.0	69.4	68.0	75.2
Humaneval+	65.9	63.4	61.6	71.3
MBPP+	71.0	71.9	68.8	71.4

90.0)，在 Alignbench 测试中高出 0.31 分（8.22 对 7.91）。在非思考模式下，TeleChat2.5-115B 在 MATH500 测试中以 12.0 分的优势（87.0 对 75.0）超过 GPT-4o-1120，并在 BFCL 测试中有 4.74 分的优势（83.39 对 78.65）。TeleChat2.5-35B 变体在同类大小的替代品中也保持竞争力。相比于 Deepseek-R1-Qwen32B-distill，TeleChat2.5-35B 在 IFEval 测试中高出 5.67 分（78.26 对 73.33），在 BFCL 测试中高出 3.97 分（80.11 对 76.14），展现出更强的思考模式表现。

Table 6: 在思考/非思考模式下，与其他具有可比参数大小的模型中，T1-35B、TeleChat2-35B、TeleChat2.5-35B 及其他模型的比较。

Benchmark	MATH500	Alignbench	IFEval	BFCL
Thinking				
T1-35B	90.0	7.93	78.26	80.11
Deepseek-R1-Qwen32B-distill	94.3	7.42	73.33	76.14
QWQ-32B	96.0	7.97	80.09	83.10
Qwen3-32B	93.0	8.27	85.92	86.82
Non-Thinking				
TeleChat2-35B	61.0	6.97	77.74	75.32
TeleChat2.5-35B	77.0	7.74	78.52	78.28
Qwen2.5-32B	82.0	7.39	79.44	82.11
Qwen3-32B(non-thinking)	83.0	8.23	84.07	81.84

Table 7: 在思考/非思考模式下，T1-115B、TeleChat2.5-115B、TeleChat2-115B 与其他模型的比较。

Benchmark	# Params	MATH500	Alignbench	IFEval	BFCL
Thinking					
T1-115B	115B	94.0	8.22	80.15	83.39
OpenAI o1-mini	Unknown	90.0	7.91	79.07	-
Deepseek-R1	671B(A37B)	97.2	8.43	83.70	88.68
Non-Thinking					
TeleChat2-115B	115B	72.0	7.76	79.25	77.47
TeleChat2.5-115B	115B	87.0	7.94	80.93	83.39
Qwen2.5-72B	72B	82.0	7.62	83.70	79.15
GPT-4o-1120	Unknown	75.0	7.49	80.18	78.65
Deepseek-V3	671B(A37B)	90.2	8.06	86.10	77.66

4 结论

TeleChat2 、TeleChat2.5 和 T1 系列的引入代表了大语言模型（LLM）开发的一项重大进展。尽管在架构上变动不大，这些模型通过在预训练和后训练阶段的系统性升级，实现了显著的性能提升。通过公开这些具有可扩展参数配置（35B 和 115B）的模型，我们赋能研究人员和开发者利用尖端的 LLM 进行多种应用，促进自然语言处理、代码生成和推理领域的创新。这项工作不仅填补了之前研究的关键空白，还为未来关于大规模模型优化和任务特定适应的研究提供了稳固的基础。

References

Hongjun An, Wenhan Hu, Sida Huang, Siqi Huang, Ruanjun Li, Yuanzhi Liang, Jiawei Shao, Yiliang Song, Zihan Wang, Cheng Yuan, Chi Zhang, Hongyuan Zhang, Wenhao Zhuang, and Xuelong Li. Ai flow: Perspectives, scenarios, and approaches, 2025. URL https://arxiv.org/abs/2506.12479.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset, 2023. URL https://arxiv.org/abs/2305.12524.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, and et al. Angela Fan. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

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

Jian Hu, Jason Klein Liu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models, 2025. URL https://arxiv.org/abs/2501.03262.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019. URL https://arxiv.org/abs/1811.06965.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. arXiv preprint arXiv:2305.08322, 2023.

"Teknium" "interstellarninja". Hermes-function-calling-dataset-v1. URL https://huggingface.co/NousResearch/hermes-function-calling-v1.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback, 2024. URL https://arxiv.org/abs/2406.09279.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Chenliang Li, Hehong Chen, Mingshi Yan, Weizhou Shen, Haiyang Xu, Zhikai Wu, Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, Hongzhu Shi, Ji Zhang, Fei Huang, and Jingren Zhou. Modelscope-agent: Building your customizable agent system with open-source large language models. ArXiv, abs/2309.00986, 2023a. URL https://api.semanticscholar.org/CorpusID:261531214.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023a. URL https://arxiv.org/abs/2310.01889.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023b. URL https://arxiv.org/abs/2305.01210.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Zhuoer Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, Xiaohan Zhang, Lichao Sun, Xiaotao Gu, Hongning Wang, Jing Zhang, Minlie Huang, Yuxiao Dong, and Jie Tang. Alignbench: Benchmarking chinese alignment of large language models, 2024a. URL https://arxiv.org/abs/2311.18743.

Xiaoran Liu, Hang Yan, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of roPE-based extrapolation. In The Twelfth International Conference on Learning Representations, 2024b. URL https://openreview.net/forum?id=JO7k0SJ5V6.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

MindSpore. Mindspore: Advanced ai framework. https://www.mindspore.cn/, 2025. Accessed: 5 Feb. 2025.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021. URL https://arxiv.org/abs/2104.04473.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155, 2022.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization, 2024. URL https://arxiv.org/abs/2404.19733.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph 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.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. Tool learning with foundation models. arXiv preprint arXiv:2304.08354, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. arXiv preprint arXiv:2305.18290, 2023.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. arXiv preprint arXiv:1910.02054, 2020.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Jiawei Shao and Xuelong Li. Ai flow at the network edge, 2024. URL https://arxiv.org/abs/2411.12469.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noam Shazeer. Glu variants improve transformer. arXiv preprint arXiv:2002.05202, 2020.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864, 2022.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge, 2019. URL https://arxiv.org/abs/1811.00937.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. ArXiv, abs/2402.10176, 2024. URL https://api.semanticscholar.org/CorpusID:267681752.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024a. URL https://arxiv.org/abs/2406.01574.

Zihan Wang, Xinzhang Liu, Shixuan Liu, Yitong Yao, Yuyao Huang, Xuelong Li, Yongxiang Li, Zhonghao Che, Zhaoxi Zhang, Yan Wang, Xin Wang, Luwen Pu, Huinan Xu, Ruiyu Fang, Yu Zhao, Jie Zhang, Xiaomeng Huang, Zhilong Lu, Jiaxin Peng, Wenjun Zheng, Shiquan Wang, Bingkai Yang, Xuewei he, Zhuoru Jiang, Qiyi Xie, Yanhan Zhang, Zhongqiu Li, Lingling Shi, Weiwei Fu, Yin Zhang, Zilu Huang, Sishi Xiong, Yuxiang Zhang, Chao Wang, and Shuangyong Song. Telechat technical report, 2024b. URL https://arxiv.org/abs/2401.03804.

Mingyu Xu, Xin Men, Bingning Wang, Qingyu Zhang, Hongyu Lin, Xianpei Han, and weipeng chen. Base of roPE bounds context length. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=EiIelh2t7S.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. arXiv preprint arXiv:1910.07467, 2019.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Manoj Awalgaonkar, Rithesh Murthy, Eric Hu, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. xlam: A family of large action models to empower ai agent systems. ArXiv, abs/2409.03215, 2024a. URL https://api.semanticscholar.org/CorpusID:272424184.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on gaokao benchmark, 2024b. URL https://arxiv.org/abs/2305.12474.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. Ape210k: A large-scale and template-rich dataset of math word problems, 2020. URL https://arxiv.org/abs/2009.11506.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023. URL https://arxiv.org/abs/2304.11277.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.