MiniCPM4： 终端设备上超高效的大型语言模型

MiniCPM Team

https://huggingface.co/openbmb/MiniCPM4-8B
https://huggingface.co/openbmb/MiniCPM4-0.5B
https://github.com/openbmb/minicpm

Abstract

本文介绍了一个高度高效的大型语言模型（LLM）MiniCPM4，专为终端设备设计。我们通过在四个关键维度上的系统创新实现这种效率：模型架构、训练数据、训练算法和推理系统。具体而言，在模型架构方面，我们提出了 InfLLM v2，一种可训练的稀疏注意力机制，加速了长上下文处理的预填充和解码阶段。在训练数据方面，我们提出了 UltraClean，一种高效且准确的预训练数据过滤和生成策略，以及 UltraChat v2，一个全面的监督微调数据集。这些数据集使得模型通过仅使用 8 万亿训练标记即可实现令人满意的性能。在训练算法方面，我们提出了 ModelTunnel v2，用于高效的预训练策略搜索，并通过引入块滚动的负载平衡强化学习和高效数据的三元 LLM BitCPM，改进现有的后训练方法。在推理系统方面，我们提出了 CPM.cu，它集成了稀疏注意力、模型量化和推测抽样，以实现高效的预填充和解码。为满足多样化的设备需求，MiniCPM4 提供了两个版本，分别具有 0.5B 和 8B 参数。充分的评估结果表明，MiniCPM4 在多个基准测试中，超越了类似规模的开源模型，突显其效率和有效性。值得注意的是，在处理长序列时，MiniCPM4 -8B 相比 Qwen3-8B 表现出显著的速度提升。通过进一步的适应性调整，MiniCPM4 成功支持多种应用，包括可靠的调查生成和基于模型上下文协议的工具使用，清晰地展现了其广泛的可用性。

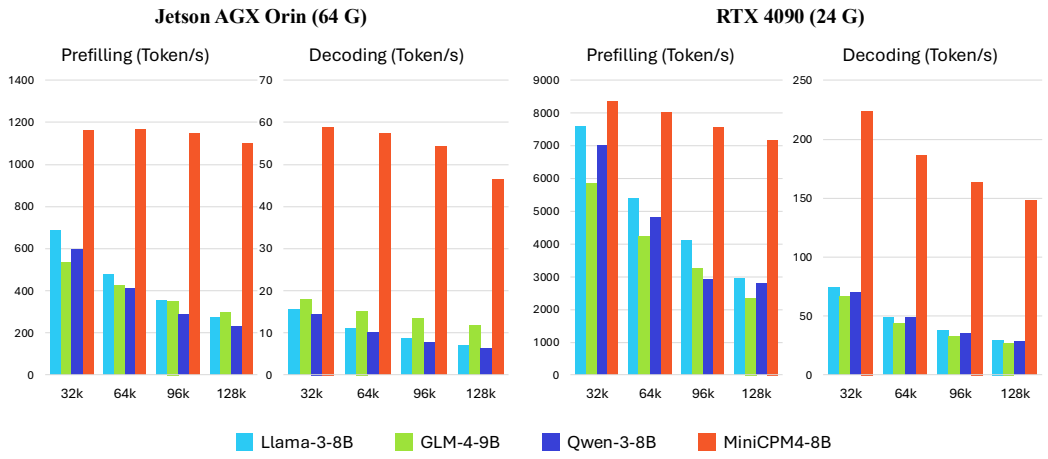

Figure 1: 终端 GPU 上的推理速度评估。

Contents

1 介绍

大型语言模型（LLMs）（Brown et al., 2020; OpenAI, 2023），也称为基础模型（Bommasani et al., 2021），已成为人工智能（AI）（Qiu et al., 2020; Han et al., 2021）领域的核心驱动力。这些大型模型展现出了处理多样化任务的卓越能力，从帮助型聊天机器人系统（Ouyang et al., 2022）到复杂的推理系统（OpenAI, 2024b; DeepSeek et al., 2025），显著提升了人机交互的质量和效率。然而，随着模型规模继续扩大（Kaplan et al., 2020; Hoffmann et al., 2022），对计算资源的需求呈指数增长，导致这些模型主要部署在云服务器上并通过 API 接口访问。

目前，LLM 的开发正面临小型化和增效的重要趋势。从 LLM 应用的角度看，高效模型可以降低部署成本，并扩展应用场景，尤其是在计算资源有限的环境中，如终端设备和移动终端（Gunter et al., 2024; OpenAI, 2024a）。从技术发展的角度看，随着模型规模的不断扩大，提高计算效率对于在有限资源情况下克服性能瓶颈变得至关重要（DeepSeek et al., 2024）。因此，在保持模型能力的同时，尽量减少计算需求的高效模型架构和算法具有相当大的理论和实际意义。

为了与更高效的大规模语言模型（LLM）趋势保持一致，我们团队一直专注于构建高效的终端侧 MiniCPM 模型（Hu et al., 2024; Yao et al., 2024）。在本文中，我们通过在模型架构、训练数据、训练算法和推理系统这四个关键维度上的系统创新，进一步提高了模型效率。基于这些进展，我们成功开发了 MiniCPM4，一个能够在终端侧芯片上高效运算的 8 B LLM。值得注意的是，相较于高效 LLM Qwen3-8B，MiniCPM4 仅使用其 22% 的训练数据便实现了相当的性能，同时在处理 128K 长度文档时，展示了在终端侧设备上 7 倍的速度提升。

具体来说，MiniCPM4 具备以下技术，以提高计算效率的同时保持模型能力：

模型架构：可训练的稀疏注意力　随着 LLMs 在长上下文处理（OpenAI, 2025; Jimenez et al., 2023）和深度推理能力驱动中的广泛应用（DeepSeek et al., 2025; OpenAI, 2024b），对 LLMs 理解和生成长序列的需求变得越来越关键。然而，自注意力机制的计算和内存需求对在终端设备上高效处理长文档构成了重大挑战。我们提出了一种稀疏注意力架构，在保持模型性能的同时，支持高效的长上下文处理。

- InfLLM v2 – 可训练的稀疏注意力，能够实现预填充和解码加速：基于我们的动态稀疏注意力架构 InfLLM（Xiao et al., 2024b），我们引入了 InfLLM v2，该版本具有高效的内核设计和端到端的专门训练。其内核在查询级别上促进了令牌级的稀疏注意力计算，从而在长上下文的预填充和解码阶段均实现了显著的速度提升。此外，我们开发了一个专门的训练框架，进一步增强了注意力机制的稀疏性并改善了长上下文处理能力。

训练数据：通过高质量数据提高效率训练　高质量的数据对于增强 LLMs 的能力密度至关重要（Xiao et al., 2024a）。虽然庞大的互联网语料库提供了丰富的训练信号，但它们不可避免地包含导致性能欠佳的噪音。基于现有的数据清洗和过滤方法（Penedo et al., 2024），我们引入了一种高效的迭代数据清洗策略，UltraClean，从而获得了 UltraFineWeb（Wang et al., 2025b），一个高质量的知识密集型数据集。此外，鉴于推理密集型数据的稀缺性，我们专门为数学和编程进行大规模数据合成。这些方法使我们能够在仅使用 8 万亿令牌的预训练数据的情况下开发出性能令人满意的模型。

- UltraClean——高质量预训练数据过滤：我们开发了一种高效且有效的数据过滤策略，名为 UltraClean，它具有高效的验证策略和高效的质量分类器。具体而言，与传统方法通过使用候选语料库从头训练 LLM 来验证数据质量相比，我们提出的高效验证策略利用一个接近训练完成的 LLM 作为基础。在最后的训练步骤中，我们引入候选语料库，并使用由此产生的性能改进作为评估数据质量的指标。这种验证策略显著提高了评估效率，同时保持了质量评估的准确性。基于我们高效的验证策略，我们可以公平地选择高质量的种子数据进行分类器训练。基于“高质量的种子数据对 LLM 训练有益”的假设，我们开发并优化了选择分类器训练种子和方案的策略，同时精心挑选平衡的正负样本集，以确保分类器的质量和鲁棒性。我们将所提出的数据过滤流程应用于 FineWeb（Penedo et al., 2024）和 Chinese FineWeb（Yu et al., 2025b）数据集，统称为 UltraFineWeb。该流程不仅提高了过滤效率、分类器质量和鲁棒性，还显著降低了实验和推理成本。
- UltraChat v2——高质量监督微调数据生成：基于 UltraChat（Ding et al., 2023），我们引入了一种高质量对话构建策略，使得能够高效生成用于大型语言模型训练和评估的推理密集型数据。具体来说，与传统的以覆盖面或表面多样性为优先的指令微调数据集相比，我们提出的策略更加注重具有深层推理、上下文一致性和任务复杂性的多轮交互。我们结合专家模型和提示工程，制作出在多种推理类型（包括多跳推理、常识推理和领域特定问题解决）上对大型语言模型具有挑战性的对话。这种方法不仅提高了数据质量，还支持创建用于微调和评估的健壮且具有挑战性的基准。基于这种推理密集的生成流程，我们构建了高质量的种子数据，并优化了提示、对话结构和质量控制机制的设计。我们进一步采用了结合自动验证和选择性人工审核的双阶段过滤过程，以确保响

应的准确性、一致性和多样性。我们利用提议的流程生成 UltraChat v2，这是对现有指令微调语料库的推理密集型扩展。该流程不仅提高了数据生成质量和过滤效率，还显著增强了在此数据上微调的 LLM 的推理能力。

训练算法：多维训练优化策略　大型语言模型 (LLM) 的缩放定律表明，随着训练量的增加，性能会提高 (Kaplan et al., 2020) 。因此，降低训练成本对于可持续的模型扩展至关重要。在 Hu et al. (2024) 中，我们开发了 ModelTunnel，通过对小规模模型进行一系列实验来寻找最佳的训练策略。在开发 MiniCPM4 的过程中，我们进一步提高了搜索精度，并引入了 ModelTunnel v2。为了实现高效的后训练机制，我们在强化学习 (RL) 过程中采用了负载均衡的展开策略，这可以在展开过程中更好地利用计算资源。此外，为了减少存储需求，我们引入了一种三元 LLM，BitCPM4。

- ModelTunnel v2 ——高效训练策略搜索：继我们的 ModelTunnel (Hu et al., 2024) 之后，我们开发了 ModelTunnel v2，通过以下两个方面进行了改进：(1) 性能指标的提升：由于出现了新的能力 (Wei et al., 2022) ，之前可预测的扩展方法无法有效预测下游任务的表现。在 MiniCPM4 中，我们构建了 ScalingBench (Xiao et al., 2024a) ，并建立了 ScalingBench 的损失与下游表现之间的关系。因此，可以使用 ScalingBench 作为性能指标来代替语言模型损失，从而提高超参数搜索的有效性。(2) 搜索有效性验证：借助 ScalingBench (Xiao et al., 2024a) ，我们系统地验证了识别参数的有效性。我们的实验结果表明，最大更新参数化（μP） (Yang et al., 2022) 与我们的超参数搜索相结合，能够实现与业界先进方法相媲美的性能，并且我们的方法可以显著降低搜索成本。
- 块状展开——负载均衡强化学习：我们实施了一种块状展开策略以提高 RL 训练效率，该策略限制了每个展开阶段的最大输出标记预算，并在后续迭代中恢复不完整轨迹的生成，显著减少了冗长轨迹导致的闲置计算。此外，为了处理块状展开策略引入的不稳定性，我们结合了几种稳定化技术，包括 KL 损失、双重剪裁、块级重要性采样和乱码过滤。这些增强措施共同确保了长链式思维 (CoT) RL 训练的稳定且高效的扩展。
- BitCPM4 – 三值大模型的量化感知训练：我们设计了一个两阶段的训练框架，通过使用我们预训练的高精度模型来初始化量化阶段，大幅降低了量化感知训练（QAT）的成本。与 ModelTunnel v2 结合，我们的方法在使用少 10 × 训练标记的情况下，实现了与现有 QAT 方法相当的性能。对于那些极其资源有限的设备，我们将 MiniCPM4 适配为三值版本 BitCPM4，并显示出良好的结果。
- 高效训练工程：受 DeepSeek et al. (2024) 的启发，我们实现了多标记预测训练目标 (Gloeckle et al., 2024) 和 FP-8 混合精度训练框架。多标记预测可以引入更多的密集监督信号，并使附加头在猜测采样中达到更高的接受长度。FP-8 混合精度训练可以充分利用我们 GPU 集群的计算能力。

推理系统：高性能推理框架用于终端设备　终端设备计算能力和存储资源有限。为了充分利用这些设备，我们定制了一个高性能推理框架。

- CPM.cu – 轻量高效的 CUDA 推理框架：我们首先开发了一个轻量级推理框架，特点是静态内存管理、内核融合和高效的推测性采样实现，从而实现了高效的预填充和解码速度。在此框架的基础上，集成了适用于 InfLLM v2 的高效稀疏注意力内核，通过引入 FR-Spec (Zhao et al., 2025) 进一步提高推测性草稿速度，引入了更加有效的前缀感知量化方法 P-GPTQ，并通过 SpecMQuant (Zhang et al., 2025b) 研究推测性采样和量化的结合效果。
- ArkInfer – 跨平台部署框架：为了应对在不同硬件平台上部署 LLMs 的挑战，我们设计了 ArkInfer，其具有统一的基于执行器的架构和自适应的后端接口。我们通过标准化 API 集成了多个推理框架 (NeuroPilot、Genie、RK-LLM、TensorRT-LLM 和 llama.cpp)，并采用了高级优化技术，如典型的推测采样和量化方法。此设计能够实现具备本机多模态功能、灵活采样策略和全面性能评估工具的无缝跨平台部署，大大简化了 MiniCPM 模型在超越 NVIDIA 芯片的各种终端设备上的集成。

基于上述技术，我们构建了 MiniCPM4 ，有两个参数版本：0.5B 和 8B，每个版本都有普通和深度推理变体。在预训练过程中，我们在 8.3T 高质量标记上训练 8B 模型。我们采用了热身-稳定-衰减 (WSD) 学习率调度器 (Hu et al., 2024) ，为热身和稳定阶段分配了 7T 标记，为退火阶段分配了 1.3T 标记。在此之后，我们进行长上下文预训练，以将 MiniCPM4 的上下文窗口从 4K 扩展到 128K 标记。随后，我们进行监督微调 (SFT) 后训练，使模型能够遵循用户指令。为了进一步开发深度推理模型，即 MiniCPM4 -Reasoning，我们利用长 CoT 数据进行 SFT，并实现含数学和编码任务的强化学习。我们在一系列广泛使用的基准上评估 MiniCPM4 。MiniCPM4 在具有相似参数规模的典型基线中表现优于其他，并成为最有效和高效的开源大语言模型之一。

最后，基于 MiniCPM4 ，我们开发了三个应用来展示 MiniCPM4 的有效性并探索先进技术，包括可信任调查生成和使用具有模型上下文协议 (MCP) 的工具。所有三个应用都要求模型能够生成具有高连贯性和逻辑性的长序列，调用复杂功能以获取外部资源，并进行创造性写作。结果表明，MiniCPM4 在这些应用上表现出有希望的结果，我们鼓励社区基于我们的 MiniCPM4 探索更多有趣的应用。

在本节中，我们介绍了 MiniCPM4 中应用的模型架构和训练算法，该模型具有高效的稀疏注意力层和高效的训练流程。在本节中，我们将首先介绍我们提出的 InfLLM v2 的细节，该版本支持高效的长上下文处理。具体而言，InfLLM v2 使得 MiniCPM4 能够以 81% 的注意力稀疏程度实现与全注意力机制相当的长上下文处理能力。然后，我们描述了预训练数据管理方法，以支持高效的知识学习。最后，我们介绍了我们的预训练流程，包括用于超参数搜索的 ModelTunnel v2 以及针对预训练目标和长上下文扩展的工程。这些方法使得 MiniCPM4 -8B 能够仅使用 Qwen3-8B 的 22% 的预训练 tokens 就实现与 Qwen3-8B 相当的结果。

继大多数开源 LLMs 之后，我们采用 Transformer (Vaswani et al., 2017) 作为我们的基本架构。考虑到处理长序列的新兴需求，许多努力致力于设计一种无训练的稀疏注意机制，以动态选择相关的上下文标记来进行长上下文处理 (Xiao et al., 2023; Jiang et al., 2024; Xu et al., 2025; Zhang et al., 2025a)。由于其稀疏性不佳，这些模型只能应用于预填充加速。

最近，MoBA (Lu et al., 2025) 和 NSA (Yuan et al., 2025) 在预训练阶段应用稀疏注意力来提高模型性能。然而，MoBA 采用了查询块的设计，这阻碍了其在解码阶段实现加速。此外，据我们的观察，相邻标记之间的相关上下文通常变化很大。因此，强制相邻标记共享相同的上下文可能导致次优性能，并且同时注意力的稀疏性无法提高。NSA 引入了三种不同的注意力组件来捕获长距离信息。三种注意力组件引入了额外的参数，这将导致短序列的计算开销增加以及预训练时三倍的键值存储成本。

在本节中，基于我们之前的稀疏注意力模型 InfLLM (Xiao et al., 2024b)，我们设计了一种可训练的稀疏注意力 InfLLM v2，以减少补填和解码短语时的计算和内存访问成本。InfLLM v2 不会为注意力输出引入额外的参数，也不会影响短序列的推理。此外，我们提出了一种高效的 Top-K 上下文块选择方法，与 NSA 相比，可以减少 60% 的计算成本。在接下来的段落中，我们将介绍 InfLLM v2 的算法设计。

在稀疏注意机制中，我们仅选择部分上下文标记进行注意力计算。根据 InfLLM，我们将键值缓存分成块级单元，每个查询标记将选择相关性得分最高的块进行注意力计算。形式上，在每个注意力层中，输入是一序列隐藏向量，每个向量代表一个标记的上下文信息。给定输入序列 XMATHX_x，自注意机制首先将该序列映射到查询、键和值向量 $\boldsymbol{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_l\}, \boldsymbol{K} = \{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_l\}, \boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_l\}$。这里，$l$ 是输入序列的长度。在密集注意机制中，每个标记需要关注之前的所有标记，这可以表示为 $\boldsymbol{o}_i = \text{Attention}\left(\boldsymbol{q}_i, (\boldsymbol{K}_{1:i}, \boldsymbol{V}_{1:i})\right)$。相比之下，稀疏注意机制只选择关注相关标记的子集。我们将键值缓存分块，以避免细粒度的标记级相关性计算和内存访问，从而提高稀疏注意的效率。具体而言，InfLLM v2 将键值缓存分为等大小的块，每个块包含 m 个标记。因此，键值缓存被分割成块 $\mathcal{B} = \{\boldsymbol{B}_0, \ldots, \boldsymbol{B}_{\lfloor \frac{l}{m} \rfloor - 1}\}$，其中 $\boldsymbol{B}_j = \left(\boldsymbol{K}_{jm:(j+1)m}, \boldsymbol{V}_{jm:(j+1)m}\right)$。这里，$m$ 是键值块的块大小。

InfLLM v2 的稀疏注意力计算由两个阶段组成。在第一阶段，我们根据查询 token $\boldsymbol{q}_i$ 动态选择 $\mathcal{B}$ 中的相关块。为此，我们需要计算查询 token $\boldsymbol{q}_i$ 与每个块之间的相关性得分 $r_{\text{block}}(\boldsymbol{q}_i, \boldsymbol{B}_j)$，然后选择具有最高相关性得分的块。在第二阶段，基于第一阶段选择的块，我们计算 $\boldsymbol{q}_i$ 与这些选定块内所有 token 之间的注意力。在接下来的章节中，我们将介绍这两个阶段的细节。

1.0.1 动态上下文块选择

InfLLM v2 的最关键组件是查询 token 与键值块之间的相关性得分计算。为了避免逐个 token 进行相关性计算，InfLLM (Xiao et al., 2024b) 从每个块中选择代表性 token 作为块的表示，以查询 token 与这些代表性 token 的点积定义相关性得分。虽然这种方法可以捕捉块内的重要语义信息，但选择代表性 token 需要进行 token 级计算和内存访问，成为 InfLLM 效率瓶颈之一。因此，我们引入细粒度语义核以捕捉块语义并避免 token 级内存访问。此外，我们要求同一组中的查询头共享相同的键值块以减少内存访问成本。

语义核　InfLLM v2 进一步改进了相关性分数计算方法。由于稀疏注意机制需要尽量减少内存访问的不连续性，InfLLM v2 要求对键值序列进行粗粒度的块划分，其中块大小 m 通常是一个相对较大的值。如果我们仅用一个向量来表示每个块的语义，将不可避免地遇到信息损失的问题。为了实现更准确的相关性计算，InfLLM v2 引入了细粒度的语义核来构建每个键值块的表示。具体来说，InfLLM v2 在更细的粒度上划分键值序列，产生多个语义核。为了确保输入序列中的每个语义跨度都包含在一个完整的语义核内，这些核需要互相重叠。形式上，InfLLM v2 将输入序列划分为大小为 p 的语义核，相邻核之间的步幅为 s。因此，$\boldsymbol{K}$ 被划分为 $\mathcal{S} = \{\boldsymbol{S}_0, \ldots, \boldsymbol{S}_{\lfloor \frac{l}{s} \rfloor - 1}\}$，其中 $\boldsymbol{S}_{\hat{j}} = \boldsymbol{K}_{\hat{j}s:\hat{j}s+p}$。由于语义核仅参与相关性计算，因此只需保留键向量。

InfLLM v2 使用均值池化操作来计算每个语义核 $\text{Mean}(\boldsymbol{K}_{\hat{j}s:\hat{j}s+p})$ 的表示，查询令牌 $\boldsymbol{q_i}$ 与语义核之间的相关性评分定义为 $r_{\text{kernel}}(\boldsymbol{q_i}, \boldsymbol{S}_j) = \text{softmax}(\boldsymbol{q_i} \cdot \text{Mean}(\boldsymbol{K}_{\hat{j}s:\hat{j}s+p}))$。查询令牌 $\boldsymbol{q_i}$ 与块 $\boldsymbol{B}_j$ 之间的

相关性评分可以表示为与 $\boldsymbol{B}_j$ 相交的所有语义核的最大相关性评分：

$$r_{\text{block}}(\boldsymbol{q_i}, \boldsymbol{B}_j) = \max r_{\text{kernel}}(\boldsymbol{q_i}, \boldsymbol{S}_{\hat{j}}), \quad \boldsymbol{S}_{\hat{j}} \in \mathcal{S} \quad \text{and} \quad \boldsymbol{B}_j \cap \boldsymbol{S}_{\hat{j}} \neq \emptyset. \tag{1}$$

基于相关性评分 $r_{\text{block}}(\boldsymbol{q_i}, \boldsymbol{B}_j)$，InfLLM v2 选择相关性最高的 k 个块。然后，InfLLM v2 计算查询令牌 $\boldsymbol{q_i}$ 与这些选定块内所有令牌之间的注意力，以生成最终输出 $\boldsymbol{o_i}$。

值得注意的是，考虑到初始标记以及局部窗口内的标记通常对最终输出贡献颇多，InfLLM v2 将每个查询标记 $\boldsymbol{q_i}$ 与初始键值块 $\boldsymbol{B}_0$ 以及局部窗口块之间的相关性分数设为无穷。该机制确保每个查询标记都能访问初始块和局部窗口内的块。当文本长度较短且不超过 k 块的总长度时，InfLLM v2 退化为普通的密集注意机制。

Top-K 块共享　当前的大型语言模型（LLM）架构通常采用分组查询注意力层，其中多个查询共享一个键值头。在 InfLLM v2 中，我们要求同一组中的查询头共享相同的 Top-K 相关块。这使我们能够尽可能减少内存访问。具体来说，在每个查询头用语义核计算相关性得分后，我们在组内对相关性得分进行平均，并将此平均值用作查询组的相关性得分。

高效的 Top-K 实现　。Top-K 选择涉及三个顺序步骤：1）计算查询标记和每个语义核之间的相关性得分，然后进行 softmax 归一化。2）在查询组维度上聚合每个语义核的相关性得分。3）基于聚合的得分为每个查询标记选择 Top-K 上下文块。此操作构成稀疏注意力机制中的计算瓶颈。

传统的稠密注意力机制通常采用 FlashAttention 算法（Dao et al., 2022）来降低内存使用并加速注意力计算。具体来说，FlashAttention 通过在线 softmax 减少注意力计算过程中的 HBM 访问操作。然而，Top-K 选择与注意力计算过程本质上不同，因为它需要每个查询组与每个语义核之间关联分数的精确数值。由于关联计算、softmax 归一化和跨查询组维度的分数聚合不满足交换律，在线 softmax 无法应用。因此，Top-K 选择需要对语义核进行两次内存访问和计算，其中第一次用于计算 LogSumExp (LSE)，第二次用于计算最终的关联分数。

Top-K 选择是 InfLLM v2 进行长上下文处理的瓶颈。为了降低计算成本，我们提出了一种高效的 LSE 近似方法。与计算查询标记和所有语义核之间的点积结果不同，我们尝试通过引入粗粒度的语义核来近似 LSE 值，其核大小 s_c 远大于 s。然后我们计算查询标记和粗粒度语义核之间相关性得分的 LSE。该方法的计算和内存访问成本仅为原方法的 $\frac{s}{s_c}$。

随着那些需要长上下文处理和深度推理能力的应用的发展，可训练的稀疏注意力机制显示出极大的潜力，可以提高预训练和推理的效率。在本节中，我们讨论了 InfLLM v2 的几个关键特性和设计原则。我们希望这些讨论能够促进可训练稀疏注意力机制的未来进步。

复杂性分析　InfLLM v2 使每个 token 仅与前 k 个键值块进行注意力计算，从而显著减少了注意力机制的计算和内存访问开销。在本段中，我们分析 InfLLM v2 的计算和内存访问复杂度。在第 1 阶段，我们需要计算查询 token 与每个语义核之间的相关性得分。对于一个上下文长度为 l 的查询 token，有 $\lfloor \frac{l}{s} \rfloor$ 个语义核。因此，该查询 token 需要进行 $\lfloor \frac{l}{s} \rfloor$ 次向量乘法和内存访问。在第 2 阶段，我们需要计算查询 token 与 k 个键值块之间的相关性得分。在此过程中，查询 token 需要进行 $2km$ 次向量乘法和内存访问。相比于需要 $2l$ 次向量操作和内存访问的稠密注意力机制，当序列非常长（$l \gg m$）时，InfLLM v2 可以将计算开销和内存访问减少到 $\frac{1}{s}$。我们可以看到，第 1 阶段的计算复杂度保持为 $O(l^2)$，而第 2 阶段的计算复杂度是 $O(l)$。因此，如果我们想进一步提高稀疏注意力的效率，我们应该更多地致力于相关块检索。

查询和键值标记的不同粒度　在 InfLLM v2 中，我们允许每个查询标记与不同的键值块计算注意力。这里，查询的计算单元是标记级别的，而键值的计算单元是块级别的。许多以前的稀疏注意力方法（Xu et al., 2025; Zhang et al., 2025a）也将查询序列划分为块，其中一个块内的所有查询标记共享相同的前 k 键值块。然而，查询阻塞操作只能加速长序列的预填充，但无法加速解码过程，因为解码需要逐标记生成，并且在大多数情况下，查询标记无法形成一个完整的块。因此，在训练期间对查询使用块级别的单元会导致解码时的训练-推理不一致性，从而降低模型性能。

可训练的上下文选择　在稀疏注意力机制中，top- k 选择操作是不可微的。这意味着语义核的表示不能通过阶段 2 中的稀疏注意力计算进行优化。因此，在 InfLLM v2 中，我们选择使用均值池化这一无参数操作来构建语义核的表示。此外，均值池化具有确保语义核的表示保持在与词级别键向量相同语义空间的优势。通过这种方式，我们可以通过优化词级键向量间接优化语义核的表示。NSA（Yuan et al., 2025）采用了一种压缩注意力机制，将上下文选择的输出添加到注意力输出中，从而能够优化块表示。然而，这一操作对短文本增加了显著的开销。相比之下，InfLLM v2 采用的均值池化操作不会影响短文本的效率，提供了更大的实际价值。

超参数推荐　稀疏注意力机制的高效训练和推理需要高度协调的算法设计和运算符设计。因此，从算法有效性和硬件限制的角度来看，InfLLM v2 中的超参数设计需要满足某些条件。关于语义核大小的选择，较小的核大小可以实现更精确的相关性分数计算；然而，较小的核大小也会带来更大的

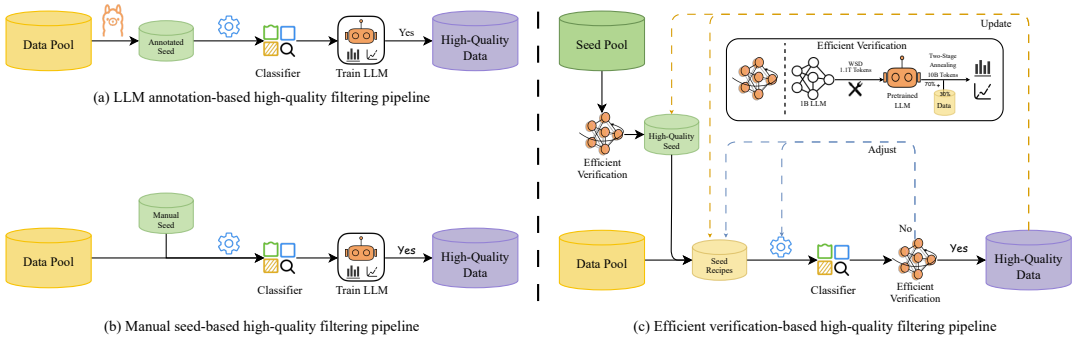

Figure 2: 高质量数据过滤管道的示意图。传统的基于模型的数据过滤方法 (a) 和 (b) 依赖于人工专业知识进行种子数据选择，并且缺乏数据质量验证。

计算开销。因此，为了在有效性和效率之间达到良好的平衡，我们选择将核大小设置为 32，步幅为 16。受 GPU 张量核的矩阵乘法累加指令的限制，一个查询组必须至少包含 16 个头部，才能确保硬件被充分利用。

1.1 UltraClean：高质量预训练数据过滤与生成

随着大型语言模型的快速发展，数据质量成为提升模型性能的关键因素之一。因此，为了增强 MiniCPM4 的能力密度，我们进行了广泛的数据工程，包括从海量互联网源中筛选高质量的知识密集型数据，并利用现有的大型语言模型生成高质量的推理密集型数据。具体来说，我们引入了 UltraClean，一种高效的预训练过滤技术，基于此，我们可以使用 8 万亿的标记达到与 Qwen3-8B 经过 36 万亿标记训练相当的性能。

1.1.1 高质量知识密集型数据过滤

随着大型语言模型（LLMs）的迅速发展，数据质量已成为提高模型性能的关键因素之一。利用基于模型的分类器来过滤数据并提取高质量的知识密集型样本，不仅可以增强模型的有效性，还能够通过在更少的 token 下实现更好的性能，降低训练成本。然而，当前的方法仍面临两个主要挑战：（1）缺乏高效的数据验证策略，难以对数据质量提供及时的反馈；（2）用于训练分类器的种子数据的选择缺乏明确标准，并且在很大程度上依赖于人工经验，这引入了主观偏差。为了解决这些问题，我们提出了一种高效的数据验证策略，能够以最小的计算成本快速评估数据对 LLM 训练的实际影响。在此基础上，我们基于高质量种子数据有助于提升 LLM 性能的假设优化正负样本选择过程，并构建了一条高效的数据过滤流程。我们的方法不仅产生了更为稳健和高质量的知识密集型分类器，同时也显著提高了预训练数据的整体质量。

总体工作流程　整体工作流程如图 2 (c) 所示。我们首先应用有效的验证策略来评估候选种子样本的初始池，选择高质量的数据，这些数据显著提高训练性能，作为分类器训练的正种子。同时，从原始数据池中随机抽取负样本以构建平衡的训练集。为了更有效地评估分类器的实际效果，我们也应用有效的验证策略来评估其过滤结果。根据验证反馈，我们迭代更新高质量种子池，动态调整正负样本的比例，并微调分类器的训练超参数，从而不断优化数据过滤策略。只有在有效验证下表现出稳定可靠性能的分类器才用于大规模数据过滤和后续模型训练。值得强调的是，最终高质量的分类器被应用于整个网络规模的预训练数据集中，以提取高质量的知识密集型训练样本，从根本上提高大规模模型训练的效率和效果。

高效验证策略　在有限的 token 预算下，LLM 训练的性能差异往往无法达到统计显著性，而训练过程的固有不稳定性进一步削弱了验证结果的可靠性。有效的预训练数据验证通常需要至少 1000 亿个 token。如表 1 所示，在具有 10 亿参数的 LLM 上训练 1000 亿个 token 大约需要 1200 个 GPU 小时，相当于 64 张 GPU 连续运行近 19 个小时。这种高计算成本使得在高质量数据分类器的迭代开发过程中进行高效验证变得不切实际。为了解决这个问题，我们从 Llama 3.1 (Dubey et al., 2024) 的设计中汲取灵感，提出了一种高效的验证策略。具体而言，我们使用 WSD 调度器 (Hu et al., 2024)，对一个 10 亿参数的 LLM 进行预训练，总覆盖 1.1 万亿（T）个 token。这包括对 1T 个 token 的稳定训练

Table 1: 在具有???B 参数的 LLM 上不同验证策略的计算成本比较。

	100B from scratch	380B from scratch	Efficient verification Strategy
GPU Hours	1,200	4,600	110

阶段和对额外 0.1T 个 token 的衰减阶段。在此基础上，我们引入了两阶段退火训练过程，其中微调是在 10B 个 token 上进行的，保留 30 % 的数据用于验证，其余 70 % 遵循默认的混合数据比例。与 1200 个 GPU 小时的完整训练成本相比，这一策略将训练时间减少到约 110 小时（即在 32 个 GPU 上少于 3.5 小时），大大降低了计算需求，并显著提高了数据过滤流程的效率和可迭代性。我们使用了一个以原始混合数据比例为比较基线的两阶段退火训练过程。该验证策略能够高效评估候选数据对模型训练在多个指标上的影响，平衡准确性和成本效益，并为数据选择过程提供了一个实用和闭环的优化路径。

分类器训练方案　目前，分类器训练的正样本选择主要依赖于 LLM 评分或人工整理。然而，前者容易受到 LLM 中固有偏见的影响，这可能会引入系统性噪声和标注伪影，而后者则严重依赖于人的专业知识，并且缺乏对种子数据有效性的严格评估。此外，人工选择的可靠性通常通过在过滤数据上训练的 LLM 的下游性能间接判断，使得验证过程既昂贵又间接。这些限制阻碍了分类器在不同任务上的适应性和泛化能力。为了解决这个问题，我们提出了一个核心假设：能够提高 LLM 性能的高质量种子数据在训练能够识别高质量训练样本的分类器时也应该是有益的。如图 2 （c）所示，我们首先将我们高效的验证策略应用于候选种子样本的初始池，并选择那些在 LLM 训练中表现出显著性能提升的样本作为分类器训练的正样本。我们评估了大量候选种子，并最终整理出一组高质量、经过实验证实的正样本。这些样本包括：LLM 评分超过 4 的高质量领域内数据、指令格式的数据集（例如，OH-2.5 和 ELI5）、真实世界的教育材料、LLM 合成的教科书式内容和精心挑选的高质量网络数据。这个过程不仅确保了正样本的整体质量，还显著提高了过滤效率，为分类器训练提供了坚实的基础。为了增强分类器的鲁棒性，我们使用来自多样化来源的原始数据构建负样本。英文负样本来自 FineWeb、C4、Dolma、The Pile 和 RedPajama，而中文负样本则包括 CCI3、ChineseWebtext 和其他主流语料库。实验结果进一步表明，结合多样化的数据源用于负样本显著提高了分类器的泛化能力和跨领域适应性。在初始训练之后，我们采用迭代训练机制：当前分类器版本推断的正负样本被用作下一轮的新训练数据，持续优化分类器性能。通过这一迭代过程，我们进一步提高了分类器在大规模数据过滤任务中的精度和稳定性。

基于 FastText 的质量过滤　目前高质量数据分类器大致可以分为基于 LLM 的方法 (Penedo et al., 2024; Yu et al., 2025b) 和基于 fastText 的方法 (Li et al., 2024b; Shao et al., 2024b) 。尽管基于 LLM 的分类器通常具有强大的性能，但它们会带来显著更高的推理成本。为了解决这个问题，我们采用了基于 fastText 的分类器，这种方法在某些条件下能够显著减少推理开销，同时保持竞争性能。该方法不仅减少了资源消耗，还加快了数据过滤实验的迭代周期。例如，使用基于 LLM 的分类器处理 15 万亿（15T）标记需要大约 6,000 小时的 GPU 时间，然而 fastText 可以在一个非 GPU 服务器上使用 80 个 CPU 在不到 1,000 小时内完成同样的任务，提供了显著的效率提升。值得注意的是，我们大部分的大规模实验是在一个分布式的 Spark 集群上进行的 [1] 。在数据预处理阶段，我们应用了一系列必要的步骤，包括去除冗余的空行和过多的空格，去除变音符号，并将英文文本规范化为小写。我们使用了 DeepSeek-V2 (Liu et al., 2024a) 的分词器，它优于传统的分词方法（例如，基于空格的英文分词和中文的 Jieba 分词 [2] ）。同时，我们保留了结构化的标记，如 `\n` 、`\t` 和 `\r` 。在训练过程中，我们将 fastText 分类器配置为以下超参数：向量维度设置为 256，学习率设置为 0.1，最大 n-gram 长度设置为 3，最小词频设置为 5，训练 epoch 数量设置为 3。在推断过程中，我们采用默认的分类阈值 0.5，以简化工作流程并确保实验中的一致性，避免额外的超参数调整。在推断过程中，我们采用默认的分类阈值 0.5，以简化工作流程并确保实验的一致性，避免额外的超参数调整。

结果与分析　在我们的实验中，我们使用 MiniCPM-1.2B 模型架构和 MiniCPM3-4B 分词器。每次实验涉及训练大约 100B 的 tokens。我们使用 Lighteval 库 (Fourrier et al., 2023) 进行模型评估，设置与 FineWeb (Penedo et al., 2024) 和 CCI3-HQ (Wang et al., 2024a) 相同。所有评估指标均基于零样本设置。如表 2 所示，在英文指标方面，UltraFineWeb-en 在多项任务上表现出显著提升，包括 MMLU、ARC-C、ARC-E、CommonSenseQA 和 OpenBookQA。具体而言，UltraFineWeb 在这些任务中均优于 FineWeb，在 HellaSwag 上相比 FineWeb 仅有 0.15 个百分点的微小下降（*pp* ），但相较于 FineWeb-edu 提升了 0.6 *pp* 。UltraFineWeb-en 的英文平均得分（45.891 *pp* ）较 FineWeb（42.287 *pp* ）高出 3.61 *pp* ，较 FineWeb-edu（44.560 *pp* ）高出 1.3 *pp* 。在中文指标方面，UltraFineWeb-zh 在 C-Eval 和 CMMLU 上也优于 FineWeb-zh 和 FineWeb-edu-zh 两个基准。具体来说，在 C-Eval 和 CMMLU 上，UltraFineWeb-zh 分别比中文 FineWeb 和中文 FineWeb-edu-v2 提高了 0.31 *pp* 和 3.65 *pp* ，并且相较

[1] https://spark.apache.org/

[2] https://pypi.org/project/jieba/

Table 2: 针对英语和中文数据集的个人结果比较。

Metrics	FineWeb	FineWeb-edu	UltraFineWeb-en
MMLU	28.84	31.80 +2.96	32.24 +3.40
ARC-C	25.17	34.56 +9.39	35.67 +10.50
ARC-E	59.18	69.95 +10.77	70.62 +11.44
CommonSenseQA	34.32	31.53 −2.79	36.45 +2.13
HellaSwag	42.91	42.17 −0.74	42.76 −0.15
OpenbookQA	22.20	25.20 +3.00	26.20 +4.00
PIQA	73.29	72.14 −1.15	73.67 +0.38
SIQA	38.95	38.13 −0.82	39.61 +0.66
Winogrande	55.64	55.56 −0.08	55.80 +0.16
Average	42.28	44.56 +2.282	45.89 +3.613
Metrics	Chinese-FineWeb	Chinese-FineWeb-edu-v2	UltraFineWeb-zh
C-Eval	33.95	34.17 +0.22	34.26 +0.31
CMMLU	32.41	34.93 +2.52	36.06 +3.65
Average	33.18	34.55 +1.37	35.16 +1.98

FineWeb-edu-zh 提高了 0.09 *pp* 和 0.13 *pp* 。UltraFineWeb-zh 的中文平均得分分别比 FineWeb-zh 和 FineWeb-edu-zh 提高了 1.98 *pp* 和 0.61 *pp* 。这些结果表明，我们提出的高质量数据过滤管道显著提高了数据质量，进而明显改善了模型性能。

在构建高性能 LLM 的预训练过程中，推理能力被广泛认为是通用智能的核心指标之一。这种能力的发展在很大程度上依赖于预训练数据的质量、结构和知识密度。然而，目前典型的预训练语料，如 Common Crawl 和大规模网络爬取的数据集，在支持复杂推理技能方面仍面临显著挑战。具体而言，在现有语料中通常观察到两个核心问题：（1）尽管网络数据具有多样的语言形式和广泛的覆盖面，但其中大部分由广告、模板文本、肤浅的对话和各种冗余信息组成。这些内容通常表现出低知识密度和弱逻辑结构。即使有高质量的数据分类器进行过滤，提取的内容仍然缺乏准确的推理链和系统化的知识组织，使得模型难以获得可迁移的思维模式和推理范式。（2）目前的数据构建过程存在明显的规模-深度权衡。用于训练 LLM 的语料高度依赖于信息密度低的长尾内容，以追求广泛的覆盖和多样性。相对而言，真正结构化和知识丰富的文本，如教科书、学术材料和系统解释性内容，仅占很小一部分。这种不平衡的数据结构限制了模型学习深层语义关系和复杂逻辑结构的能力。结果是，虽然模型可能生成流畅的语言，但在推理深度上往往缺乏，并可能在需要密集推理的任务中产生错误的推理。

为了解决这些挑战，我们提出了一种高质量、推理密集型的数据生成管道，以提高推理能力。这个任务导向且训练高效的管道整合了种子数据选择与结构化数据整理和生成。通过多轮迭代，我们逐步构建预训练数据，这些数据具有高知识密度、结构清晰、逻辑连贯且语言多样化。这不仅提高了模型在一般基准上的性能，还在基础能力层面促进了更强的推理迁移性和泛化能力。

种子数据选择　对于种子数据的选择，我们关注两个核心目标：高知识密度和领域多样性。一方面，我们利用基于 UltraFineWeb 数据集构建的高质量数据分类器，从大规模网络语料库中过滤出知识密集、逻辑完整且结构良好的内容。所选内容可以作为通用领域的高质量种子数据。这些数据片段具有较强的上下文连贯性和概念丰富性，为合成模型提供了结构化的语言模板和知识表达。另一方面，为了支持更高层次的领域特定推理能力，我们针对数学、编程和自然科学等关键领域进行人工策划。我们选择高质量的开源网络内容、教材和问答材料作为领域特定的种子。这些资源通常具有高度结构化的表达和清晰的知识组织。通过结合自动化过滤和人工策划，我们构建了一个兼顾通用性和专业化的种子池，为后续以推理为重点的数据生成奠定了坚实的基础。

结构化数据整理与生成　为了有效提高知识密度和推理能力，我们设计了一种以清晰结构、语义丰富性和逻辑递进为中心的结构化数据整理与生成机制。传统的网络语料库通常以零散和非结构化的形式呈现信息。为了解决此问题，我们采用结构优先原则，利用小于 10B 参数的开源 LLM 自动编辑和重构网络文本。关键操作包括去噪和清理、语义整合和逻辑补全，从而产生中短长度且结构完善、逻辑连贯、层次分明的段落。此外，基于高质量的开源网络语料库、教科书和问答材料，我们抽象出两种典型的知识构建范式（教科书和论坛），并制定相应的合成策略。对于教科书范式，我们强调系统性和进阶性，构建由知识点、多轮解释、总结和练习题组成的层次结构。对于论坛范式，我们通过生成围绕中心主题的多轮问答交流和观点辩论来模拟真实的用户讨论，从而增强多样性和真实

感。这些生成方法不仅保留了高知识密度，还为模型提供了多种推理路径和认知框架，促进了更为广泛和批判性思维的语言模型的发展。值得注意的是，构建的数据会反馈回原始种子池以便后续进化生成。这使得数据生成过程具有自我增强的性质，通过多轮迭代进化逐步构建更丰富、更深层次的高质量推理语料库，为模型提供持续优化的训练信号和更为坚固的认知支持结构。

1.1.2 未来训练数据的讨论

我们系统地介绍了在 MiniCPM4 的预训练中使用的高质量数据构建策略，包括高质量知识密集型数据过滤和推理密集型数据生成。在不增加甚至在某些情况下减少训练 token 的情况下，我们通过自动过滤和结构化生成显著提高了语料库的知识密度和逻辑复杂性。实验结果表明，这一策略在一系列下游任务中的性能可与在全规模语料库上训练的模型相媲美，甚至超过。在特别是知识密集型和推理密集型任务中，优化的数据更有效地增强了模型的综合能力和知识迁移能力，进一步验证了数据质量重于数据数量的原则。

尽管结果令人鼓舞，仍有一些开放性的问题需要在未来探索。首先，当前的数据过滤和生成管道仍部分依赖于人工设计的启发式方法。利用大语言模型（LLMs）的能力构建自监督机制以进行数据质量评估和结构优化是提高效率和质量的关键方向。其次，在进化生成过程中，平衡语料库的多样性与任务相关性仍然具有挑战性，需要更细粒度的反馈机制以防止语义模式崩溃。最后，将这一策略扩展到多语言、跨任务甚至多模态场景中，代表着构建下一代高性能基础模型的重要一步。

1.2 ModelTunnel v2：高效的预训练策略搜索

训练 LLMs 需要巨大的计算成本，因此如何在最大化模型性能的同时最小化计算资源消耗成为一个关键挑战。在我们之前的工作 (Hu et al., 2024) 中，我们基于可预测的扩展技术建立了一个 ModelTunnel。这使我们能够在小模型上搜索训练策略并将其转移到大模型训练上，从而减少确定大模型最佳训练配置的实验成本。在 MiniCPM4 的训练过程中，我们重用了 ModelTunnel 中的相关配置，并开发了 ModelTunnel v2，该版本提高了搜索精度，并提供了对 μP 的有效性系统验证。此外，为了提高模型训练效率，我们在预训练目标和基础设施中实施了工程优化。接下来，我们将详细描述这些改进在 MiniCPM4 预训练中的应用。

1.2.1 高效可预见的扩展与改进的性能指标

随着模型参数规模的不断增加，进行模型训练实验的成本也相应上升，单次模型训练需要数十万 GPU 小时。这使得像网格搜索这样的传统方法在进行模型训练配置搜索时越来越不切实际。在 MiniCPM-1 中，我们系统地构建了一个 ModelTunnel，在仅有数百万参数的模型上进行广泛实验以确定最佳超参数。在构建 MiniCPM4 时，我们在以下方面进行了改进：1) 更合理的性能指标 。在 MiniCPM-1 中，我们使用模型在开源预训练语料库上的语言模型损失作为性能指标，假设语言模型损失越低表明模型性能越优。然而，在开源预训练数据集上的损失并不能准确反映模型在下游任务上的表现。因此，我们构建了 ScalingBench，并建立了关于 ScalingBench 的损失与下游任务性能之间的关系（Xiao et al., 2024a）。由于在风洞中训练的大多数模型的参数和训练数据非常有限，它们往往无法在下游任务中表现出非随机性能，从而使得直接在下游任务上评估风洞中的小模型不可靠。因此，我们尝试构建一个评估数据集，通过功能映射关系使其损失与下游任务性能良好相关。为实现这一目标，我们从下游任务的验证数据集中构建 ScalingBench。在原始下游数据集中，每个实例由用户指令和通常包含几个词的人类标注标签组成。在 ScalingBench 中，我们使用 GPT-4o（OpenAI, 2023）为所有测试实例生成推理步骤。然后我们直接计算推理步骤和标签上的条件损失，具体来说，就是当模型在给定任务输入时生成答案所产生的损失。这个损失可以作为合理的性能指标。

为了验证 ScalingBench 的有效性并建立损失和下游性能之间的关系，我们评估了多个模型在 ScalingBench 上的损失和性能。这些模型由我们的团队在不同时期使用相同的词汇表进行训练，使得它们的损失具有可比性。模型的参数范围从 0.36B 到 4B，训练数据源和规模各不相同。所有模型在 ScalingBench 上均显示出相同的趋势：ScalingBench 损失和下游性能之间呈现出 sigmoid 函数关系。如图 3 所示，红色三角形代表不参与函数拟合的 7B 和 80B 参数模型，作为曲线的测试点。他们的 ScalingBench 评分和在相应任务上的表现也与 S 型关系一致。这些结果表明，我们构建的 ScalingBench 有效地建立了损失与下游任务性能之间的关系，表明在 ScalingBench 上的损失是一个非常有效的性能指标。2) μP 与基础架构 的比较。利用可预测的缩放进行超参数搜索代表了一条关键路径，以降低实验成本并最大化模型性能。这个方向最近引起了显著关注，这方面的研究大致分为两类：架构驱动的超参数迁移（Yang et al., 2022; Everett et al., 2024; Lingle, 2024; Bordelon et al.,

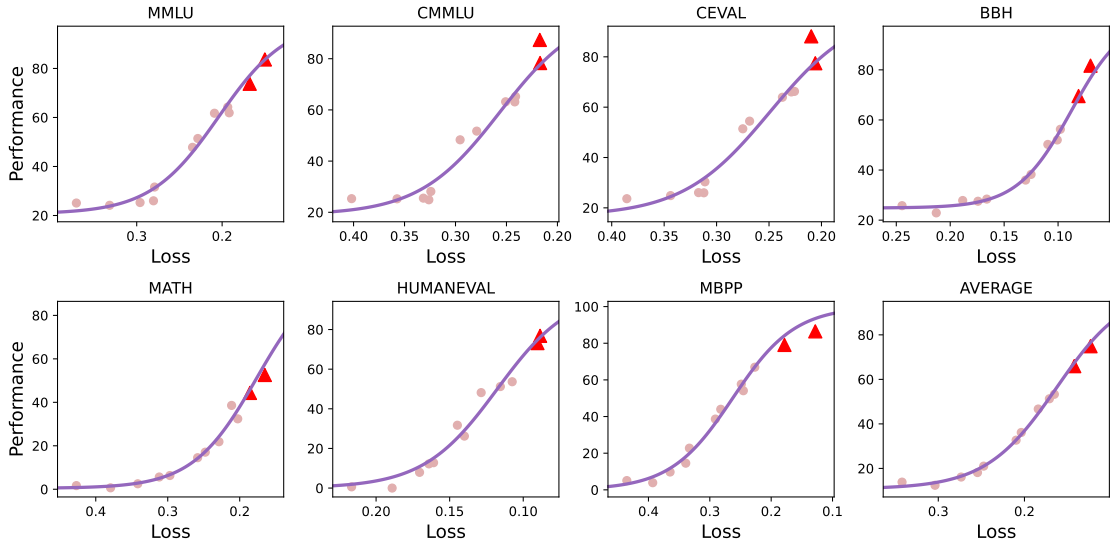

Figure 3: 在 ScalingBench 上损失与下游性能之间的 S 型关系。

Table 3: μP 和 StepLaw 之间的超参数搜索计算成本和性能比较。

# Model Params # Training Tokens		GPU Hour	150M 4B	150M 10B	150M 20B	360M 20B	360M 40B	360M 100B	700M 40B	700M 100B
LM Loss	Vallina	1M	2.1834	2.0335	1.9608	1.8510	1.7755	1.7274	1.6956	1.6391
	μP	32	2.1797	2.0491	1.9666	1.8547	1.7721	1.7174	1.7091	1.6419
ScalingBench	Vallina	1M	0.4310	0.3897	0.3685	0.3383	0.3172	0.3004	0.2984	0.2805
	μP	32	0.4434	0.3915	0.3677	0.3371	0.3196	0.3045	0.3013	0.2789

2023) 和数据驱动的超参数迁移 (Kaplan et al., 2020; Bjorck et al., 2024; Li et al., 2025a) 。前者修改模型的计算过程，以确保超参数设置可以在小型和大型模型之间共享。后者通过分析超参数与模型参数规模之间的关系来确定 LLM 的最佳超参数配置。在 MiniCPM 系列模型中，我们采用 μP (Yang et al., 2022) 作为基本架构，该架构假设在某些约束下，学习率等超参数可以在不同模型尺寸之间进行迁移。而大多数现有的开源模型采用数据驱动的方法进行超参数搜索。在 MiniCPM4 中，我们将 μP 架构与基础模型格式进行了比较。最近，已经提出了多种超参数搜索方法，其中 StepLaw (Li et al., 2025a) 是一种显著的方法。然而，我们的分析显示原始 StepLaw 研究中的实验配置与我们的实际训练需求之间存在显著差异。鉴于 StepLaw 和 μP 在超参数优化方面的潜在优势，我们希望系统地评估它们在我们特定训练环境中的相对表现。为此，我们设计了一项全面的比较研究，比较 StepLaw 和 μP 方法。

我们通过 StepLaw 来计算预测的超参数开展实验，特别关注于批大小和学习率的预测。然后，通过选择与 StepLaw 预测的批大小最接近的 16 倍值来实施批大小选择，以确保计算效率。我们比较两种不同的训练配置：一种是不使用 μP 的标准架构，并采用 StepLaw 预测的学习率；另一种是使用 μP 架构，并实证优化学习率参数。我们通过训练损失指标和 ScalingBench 评估两个训练模型的性能，对每种方法在特定训练背景下的有效性进行全面评估。如表 3 所示，StepLaw 显示了稍多的优势实例，但两种方法之间的损失值和 ScalingBench 得分差异仍然很小，没有一种方法表现出稳定的优越性。我们将 μP 框架中超参数搜索与 StepLaw 的可比有效性在实验条件下归因于以下因素：1）实际硬件限制阻碍严格遵循 StepLaw 规定的批大小。2）实验采用 WSD 学习率调度，稳定和衰减阶段使用不同的数据分配，而 StepLaw 实验利用余弦衰减。3）我们在模型评估中应用 ScalingBench，而 StepLaw 的拟合过程依赖于训练损失。此外，随机性存在于训练过程和评估指标中。我们进一步指出，再现 steplaw 的工作会产生显著的费用，而 μP 搜索所需的 GPU 时间非常少。这使普通研究人员也能使用它。总之，我们认为，使用超参数搜索方法的 μP 可以利用真实的训练配置和数据，在新的实验中进行低成本的超参数搜索。Steplaw 在实验环境中表现依然强劲，即使环境变化多样，在很多情况下仍可作为参数搜索方法的对比标准。

1.2.2 预训练工程

为了提高模型训练效率，受 DeepSeek et al. (2024) 启发，我们采用多标记预测 (Gloeckle et al., 2024) 作为我们的训练目标，以引入更密集的监督信号并提高数据效率。对于训练基础设施，我们实现了一个 FP8 混合精度计算框架。

多标记预测　传统的 LLM 预训练通常采用下一个标记预测作为训练目标，这要求模型根据先前的上下文预测下一个标记。多标记预测 (MTP) 要求 LLM 输出具有附加预测头的多个标记。在这里，附加预测头是一个一层的 Transformer (Vaswani et al., 2017)，其嵌入层和输出头与主模型共享。具体来说，给定输入标记 $\{x_0, x_1, ..., x_{l-1}\}$，主模型将生成一系列隐藏向量 $\boldsymbol{H} = \{\boldsymbol{h}_0, \boldsymbol{h}_1, ..., \boldsymbol{h}_{l-1}\}$。然后下一个标记预测目标可以计算如下：

$$\mathcal{L}_{\text{NTP}} = \frac{1}{l} \sum_i \text{CrossEntropy}\left(\text{OutputHead}(\boldsymbol{h}_i), x_{i+1}\right). \tag{2}$$

然后附加预测头的输入是隐藏向量与下一个标记的嵌入的串联：

$$\hat{\boldsymbol{h}}_i = \text{Concat}(\text{Norm}(\boldsymbol{h}_i), \text{Norm}(\text{Emb}(x_{i+1}))). \tag{3}$$

然后 MTP 训练目标可以定义为：

$$\{\boldsymbol{h}_0^{\text{MTP}}, \boldsymbol{h}_1^{\text{MTP}}, ..., \boldsymbol{h}_{l-2}^{\text{MTP}}\} = \text{Transformer}(\text{Linear}(\{\hat{\boldsymbol{h}}_0, \hat{\boldsymbol{h}}_1, ..., \hat{\boldsymbol{h}}_{l-2}\})), \tag{4}$$

$$\mathcal{L}_{\text{MTP}} = \frac{1}{l-1} \sum_i \text{CrossEntropy}\left(\text{OutputHead}(\boldsymbol{h}_i^{\text{MTP}}, x_{i+2})\right). \tag{5}$$

最终预训练目标是这两个训练目标的加权和：$\mathcal{L} = \mathcal{L}_{\text{NTP}} + \lambda \mathcal{L}_{\text{MTP}}$。

FP8 混合精度训练　鉴于 NVIDIA 的 Tensor Core GPU 具有强大的 FP8 计算能力，我们在训练中采用 FP8 混合精度。根据 DeepSeek et al. (2024)，我们对参数和激活量应用在线块状 FP8 量化，使用 128Œ128 的块大小用于参数，128Œ1 用于激活量。在量化之后，我们采用 FP8 矩阵乘法累加（MMA）指令，并使用 FP32 作为累加器的精度。为了避免 FP8 带来的不稳定性，我们仅在线性投影中应用 FP8。具体来说，我们只在前向传播中计算激活量时和反向传播中计算输入梯度时使用 FP8。考虑到参数对精度极为敏感，参数梯度以 BF16 格式计算。

在本节中，我们介绍了在预训练阶段之后对大型模型进行后训练的方法，使它们能够利用在预训练期间获取的世界知识来遵循用户指令。在监督微调阶段，我们构建了一套多样化和全面的指令，以充分激活 LLM 的能力，确保其具备强大的基础能力。为此，继我们之前的工作 (Ding et al., 2023) 之后，我们构建了 UltraChat v2，这是一个大规模的 SFT 数据集，涵盖的能力包括知识应用、推理、工具使用以及长上下文处理。

此外，为了增强模型的深度推理能力，我们采用了结合长链式推理的监督微调并集成了强化学习技术。RL 的展开过程通常面临负载不平衡的挑战，这会导致计算效率非常低。为了解决这个问题，我们提出了一种负载平衡的 RL 策略——分块展开。

在下文中，我们将详细介绍用于增强基础能力和推理能力的监督微调和强化学习技术。

1.3 UltraChat v2: 基础能力增强型 SFT 数据生成

为了提高大型语言模型的核心能力，我们设计了一个专注于任务导向能力的合成数据生成框架。在关键能力维度的指导下，该框架生成高质量的问答风格数据，涵盖广泛的技能范围，在训练后期提供更有针对性和结构化的信号。我们系统地开发了涵盖五个关键技能领域的合成数据轨道：知识应用、推理、指令遵循、长文本处理和工具使用。每个轨道都针对其目标技能的输入-输出模式和认知要求进行了量身定制，提供多样化、任务驱动且可转移的训练示例。这种方法有助于模型在关键基础能力上持续改进。

我们首先从特定领域的语料库、考试大纲和各学科的教材中提取和组织知识点，从而构建一个全面且结构良好的知识框架。基于此框架，我们利用大型语言模型生成针对各个知识点的练习问答对，从而形成知识驱动的问答数据集的初始阶段。

为了进一步增强数据的多样性和泛化能力，我们对初始实践 QA 对应用了两种进化策略：（1）指令进化，通过以多样化的方式重写提示，以模拟各种提问风格和任务形式；（2）答案多样性进化，引导模型生成合理且风格多变的答案，从而提高模型在知识理解和表达方面的稳健性。

1.3.1 推理密集型数据

推理能力是一项基本技能，使大型语言模型能够处理复杂任务并在不同领域中推广知识。与标准的问题回答任务不同，推理任务不仅仅需要事实检索。它们要求模型执行多步逻辑推理并在上下文中整合信息，同时在回复中保持连贯性和结构清晰性。为了提升大型语言模型在数学推理、逻辑建模和程序思维方面的能力，我们开发了两个专门的数据集：一个专注于数学推理，另一个专注于基于代码的推理。这些数据集旨在加强模型的推理能力，并支持更健壮且可转移的逻辑技能的发展。

数学推理数据　我们首先通过系统地分类数学知识来涵盖核心领域，包括线性代数、微积分、概率、统计、微分方程、离散数学和微分几何。这些主题进一步按照教育水平进行组织，从小学到大学，以建立一个明确的难度层次。基于这些主题，我们或者使用精心准备的种子数据，或者直接提示 LLMs 生成相关问题。然后引导模型产生答案和自我反思，以提高逻辑一致性和正确性。

在我们的数学问题生成方法中，我们专注于两个维度：指令多样性和解决路径多样性。一方面，我们将基础问题扩展为各种格式——包括选择题、填空题和开放性问题，以增强表达多样性和结构复杂性。另一方面，我们为同一问题生成多条有效的解决路径，从而扩展模型探索推理空间和推广其问题解决策略的能力。生成过程由一组启发式规则指导，例如遵循指令和控制响应长度，以确保数学的有效性和答案的可验证性。这导致生成一个具有教学价值和推理深度的数学聚焦数据子集。此外，我们实施基于难度的分层，积极减少训练期间简单问题的比例。这鼓励模型专注于中高难度的问题，有效增强其构建推理链和处理复杂逻辑结构的能力。

代码推理数据　为了增强 LLMs 的代码编程能力，我们设计了一个适用于真实开发场景的代码推理数据生成流程。首先，我们手动定义各种编码上下文、问题类别（如语义补全、错误定位和复杂逻辑理解）以及难度级别，以确保构建的任务与真实工程场景紧密相似并具有明确的推理目标。在数据构建过程中，我们从真实的 GitHub 仓库、编码挑战库或开源脚本中提取高质量的代码片段，如函数定义、算法段和类结构，作为问题生成的上下文基础。利用这些预定义的上下文和片段结构，我们使用 LLMs 生成上下文相关的代码推理问题。这些问题鼓励模型理解程序结构、模拟执行路径并进行多步骤符号推理。此外，对于每个生成的问题，我们创建了伴随的单元测试和输入输出示例，以提供可执行的验证信号，确保模型在训练期间不仅理解代码语义，还能表现出可验证的行为性能。在此基础上，我们通过将问题转换为各种格式（如输出预测、逻辑诊断和代码重写），以及引入跨语言翻译（例如，从 Python 到 Java，从 C++ 到 Rust），来进一步丰富问题的多样性。此策略不仅扩大了数据覆盖率和泛化能力，还促进了在不同语法结构和类型系统下对统一推理模式和程序语义的学习。结果是，它增强了模型的跨语言可迁移性和对真实开发任务的适应能力。

1.3.2 指令跟随数据

复杂指令的逐步构建　我们首先生成简单的基本指令，并通过增加与风格、格式和内容相关的附加需求来逐步提高其复杂性。这种自下而上的策略允许从基础任务到更复杂任务的受控进展，使模型能够更好地对指令的复杂性进行泛化。

结果可验证的指令生成　我们构建具有明确可验证约束的指令，例如长度限制、必需内容或结构要求。这些约束通过基于规则的过滤可以自动验证模型输出。通过在不同解码参数下生成多个输出，我们可以高效识别并仅保留严格满足给定条件的响应，从而支持可扩展且准确的数据生成。

增强指令多样性　为了丰富指令集的多样性，我们结合了来自多个领域和背景的提示。受 Ge et al. (2024) 中提出的方法启发，我们结合特定领域的知识与各种人格设定，生成反映更广泛现实应用的指令。

从现有数据　中反向生成指令除了从零开始生成指令之外，我们还采用了一种反向指令生成策略——将现有的高质量文本视为目标输出，并生成相应的指令，这些指令可能合理地引导到这些输出。这项技术使我们能够利用有价值的未标记语料库来扩充指令跟随数据集。

受 LongAlign (Bai et al., 2024) 的启发，我们从现有的预训练语料中构建了长上下文监督微调数据。我们从预训练语料中的各种来源（包括网页、源代码、数学内容、百科全书文本等）中抽取文档 d 。对于每个文档 d ，我们使用一个大型语言模型（LLM）生成一组 n 任务导向的查询 $Q = q_1, q_2, \ldots, q_n$ ，涵盖提取、摘要、推理以及开放域问答等多种目标。为了模拟长上下文推理，我们检索相关但可能不相关的文档以形成一个具有挑战性的长上下文输入，模拟干扰项较多的情境。对于每个查询 $q_j \in Q$ ，我们从索引的语料库中检索 k 相关文档 $d_{j,1}, d_{j,2}, \ldots, d_{j,k}$ 。然后，我们将原始文档 d 与检索到的文档连接起来形成一个扩展的上下文 $C_j = \text{Concat}(d_{j,1}, \ldots, d_{j,m-1}, d, d_{j,m}, \ldots, d_{j,k})$ ，其中 d 被插入到序列中一个随机选择的位置 $m \in 1, 2, \ldots, k+1$ 。LLM 在获得每个查询 q_j 及其上下文 C_j 时，会被要求生成一个答案 a_j 。

Algorithm 1 Chunk-wise Rollout-based Policy Optimization

Input initial policy model π_θ ; reward model R ; task prompts $\mathcal{D}$; hyperparameters $\varepsilon_{\text{low}}, \varepsilon_{\text{high}}$
Initialize replay buffer $\mathcal{R} \leftarrow \emptyset$, dynamic sampling buffer $\mathcal{B} \leftarrow \emptyset$, log-prob buffer $\mathcal{L} \leftarrow \emptyset$
for step = 1,...,M do
  Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
  Update old policy model $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
  for each $q \in \mathcal{D}_b$ do
    for $i = 1$ to G do
      Generate a chunked output $o_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$
      if o_i is unfinished: store (q, i, o_i) in replay buffer $\mathcal{R}$
    if all G outputs for q are completed:
      Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running R
      Filter out o_i and add the remaining to the dynamic sampling buffer $\mathcal{B}$ Equation (7)
  if $|\mathcal{B}| < N$: continue
  Sample a train batch $\mathcal{B}_{\text{train}} \subset \mathcal{B}$ of size N for training
  Let $\mathcal{B}_{\text{rest}} = \mathcal{B} \setminus \mathcal{B}_{\text{train}}$
  Combine $\mathcal{B}_{\text{rest}}$ and $\mathcal{R}$ for current-policy log-prob estimation
  Compute $\log \pi_\theta(o_{i,t})$ for all cached chunks in $\mathcal{B}_{\text{rest}} \cup \mathcal{R}$
  Store log-probs in log-prob buffer $\mathcal{L} \leftarrow \mathcal{L} \cup \{\log \pi_\theta(o_{i,t})\}$
  Compute token-level advantage $\hat{A}_{i,t}$ for each sample in $\mathcal{B}_{\text{train}}$
  for iteration = 1, ..., μ do
    Update policy π_θ by maximizing the objective (Equation (6))
Output π_θ

这个设计帮助模型学习定位相关内容，并在具有混合相关性的长输入中进行推理。为了确保覆盖不同的上下文长度，我们控制 C_j 的总标记数在 8K 到 64K 标记之间均匀分布。

1.3.3 工具使用数据

函数调用　该函数调用数据集结合了公开可获得的资源，如 `xlam-function-calling-60k` (Liu et al., 2024c) 和 `glaive-function-calling-v2` [3]，以及通过上下文学习生成的大量内部数据。为了确保数据质量，我们应用严格的筛选标准。具体来说，我们移除了一些样本，这些样本在真实值中调用的工具不包含在可用的工具集中，或者参数名称或类型与工具架构不一致。此外，我们在工具调用之前添加了一个思维链推理步骤。根据经验，我们发现这通过指导模型更好地理解任务并选择适当的工具和参数来提升模型性能。

代码解释器　为了提高模型在代码生成和推理方面的能力，我们构建了一系列数据示例，重点是通过代码解释器来解决问题。这包括精心挑选的开源数据集和旨在反映真实世界编码场景的内部数据。

对于开源数据集，我们使用的资源包括 CodeAct (Wang et al., 2024b) 和 Code-Feedback (Zheng et al., 2024) 。为了确保与我们的执行环境兼容，我们通过分析每段代码片段的抽象语法树 (AST) 来预处理数据，并筛选出那些导入外部包或依赖用户交互的代码。

对于内部数据，我们收集了各种类型的文件，包括 CSV、PDF、图像和视频，并提示一个大型语言模型 (LLM) 生成与每个文件内容相关的具有现实意义的代码可解问题。然后提示模型使用代码解释器解决任务，允许它在一个沙盒环境中迭代生成和执行代码。每次执行后，结果都会反馈给模型。如果模型在 10 次尝试内未能解决问题，则该数据点将被丢弃。这种方法有助于创建反馈驱动的示例，教导模型如何更有效地使用代码解决实际任务。最近的研究表明，强化学习可以增强 LLM 的深度推理能力。然而，直接将强化学习应用于端侧基模型通常会导致训练不稳定和收敛缓慢。因此，我们首先使用长链推理蒸馏数据对基模型进行有监督微调 (SFT)。这一步为模型提供了基本的推理能力，并为强化学习提供了更好的初始化。随后，我们进行强化学习，以进一步提高模型的性能。为提高训练效率，我们精心策划了训练数据，并引入了一种分块展开策略，通过优化 GPU 利用率和最小化计算浪费，显著加速了强化学习过程。

[3] https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2

1.3.4 强化学习数据整理

我们收集了大量高质量的数学和编程数据，以增强模型的推理能力。我们发现数据的质量和难度在提高模型的推理能力方面起着重要作用。

数学　我们主要从 DAPO、Deepscaler、Numina、Prime 和其他来源收集可验证的数学数据。在评估阶段，不符合预期推理格式的输出将被分配零奖励。为了确定正确性，我们采用基于规则的匹配和使用 SymPy 的符号验证相结合的方法。

代码　代码相关的数据主要来源于 LeetCode、TACO、Kodcode、Codeforces 和类似的平台。我们在 Firejail 沙箱环境 [4] 中执行 Python 代码。对于有多个测试用例的问题，奖励是根据通过的测试用例的比例来计算的，当所有测试用例都通过时，给予 1.0 的完整奖励。

数据过滤　在数据收集后，我们使用 Semhash（van Dongen & Tulkens, 2025）对强化学习（RL）训练数据和监督微调（SFT）数据进行去重。为了保留更具挑战性的样本，我们使用 DeepSeek-R1-Distill-Qwen-1.5B 为每个训练示例生成四个预测，并过滤掉那些四个预测都正确的样本。由于代码数据的数量显著少于数学数据，我们多次上采样高质量代码样本，以增加它们在训练集中的比例。

1.3.5 训练方案

基于研究界的最新进展，我们采用了一种改进版本的群体相对政策优化（GRPO）(Shao et al., 2024b)。除了原始的 GRPO 算法，我们还结合了以下改进：

动态采样　在强化学习展开阶段，我们过滤掉所有响应都正确或错误的提示，确保批次中的所有提示都能有效生成梯度，同时保持一致的批次大小。这一策略减少了梯度中的高方差，从而提高了训练的效率和稳定性。

Clip-Higher 策略　我们提高了重要性采样比率的上限截断阈值。该调整缓解了训练后期的熵塌缩，并鼓励更大的探索，帮助模型实现其潜力。

令牌级别的策略梯度损失　与在样本级别平均损失不同，我们在令牌级别计算损失。这赋予较长的序列在梯度更新中更大的权重，促使模型学习复杂的推理模式，并抑制诸如冗长和重复等不良行为，最终提高训练的稳定性和生成质量。

超长样本过滤　我们在计算损失时排除因长度限制而被截断的响应。这防止对那些被提早截断的有效推理路径进行惩罚，从而鼓励模型进行更深入的推理。

为减轻在展开阶段由于轨迹过长导致的推理吞吐量下降，我们提出了一种分块展开策略，以最大化计算资源的利用率。该策略的工作流程包括三个步骤：(1) 策略模型为所有输入样本生成固定块长度的轨迹。(2) 完成或者达到最大生成长度的轨迹用于训练。对于未完成的轨迹，计算其对数概率并存储以便后续在重要性采样中使用。(3) 未完成的轨迹与下一批新的输入合并，然后过程返回到步骤 (1)。通过采用该策略，我们显著提高了 GPU 的利用率，有效减少了在单次展开迭代中由过长输出造成的计算浪费。完整算法可以在 Algorithm 1 中找到。

由于分块展开策略将长回复在迭代过程中分解为更小的块，这可能导致在策略模型更新后部分采样轨迹的分布变动，从而影响训练的稳定性。为了应对这一挑战并确保更稳定的训练过程，我们引入了以下技术：

块级重要性抽样　由于通过块级展开策略生成的轨迹跨越多个策略模型版本，我们在块级应用重要性抽样以处理分布差异。每个块根据其来源的策略独立加权。我们发现该技术对于维持稳定和高效的策略优化至关重要。

双重剪裁　块级策略引入了部分离策略的回合，这通常会因为采样轨迹的高方差导致训练损失的突增。为了解决这个问题，我们引入了双重剪裁 (Ye et al., 2020)，它从两个方向约束策略更新范围，有效地减少了由轨迹分布差异较大引起的不稳定性。

动态参考更新的 KL 正则化与最近研究中去除 KL 损失 (Yu et al., 2025a; Xia et al., 2025) 的做法相反，我们观察到保留 KL 惩罚对于稳定训练分块式展开至关重要。为了避免过度限制策略模型的潜力，我们定期更新参考模型，以在训练稳定性和模型性能之间取得平衡。

混乱过滤器　由于分块展开策略重用了来自先前策略模型的不完整轨迹，生成损坏或不连贯文本（例如，乱码输出、过度重复）的风险增加。为了防止这些异常轨迹使训练不稳定，我们引入了一个混乱过滤器，该过滤器可以检测并从损失计算中排除这些样本。

[4] https://github.com/netblue30/firejail

基于上述技术，对于每一个具体的问题-答案对 (q, a)，行为策略 $\pi_{\theta_{\text{old}}}$ 对 G 个单独的响应 $\{o_i\}_{i=1}^{G}$ 进行采样，我们重新表述原始策略优化目标以支持我们提出的块状回滚。修改后的目标正式定义如下：

$$\mathcal{J}(\theta) = \mathcal{J}_{\text{clip}}(\theta) - \beta \cdot \mathcal{J}_{\text{KL}}(\theta), \tag{6}$$

$$\mathcal{J}_{\text{clip}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{c(t)}}(\cdot|q)}$$

$$\left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \max\left(\min\left(r_{i,t}(\theta)\cdot\hat{A}_{i,t},\ \text{clip}(r_{i,t}(\theta),\ 1-\varepsilon_{\text{low}},\ 1+\varepsilon_{\text{high}})\cdot\hat{A}_{i,t}\right),\ c\cdot\hat{A}_{i,t}\right)\right], \tag{7}$$

$$\text{s.t.} \quad 0 < \left| \{o_i \mid \texttt{is\_equivalent}(a, o_i)\} \right| < G,$$

$$\mathcal{J}_{\text{KL}}(\theta) = D_{\text{KL}}\left(\pi_\theta \,\|\, \pi_{\text{ref}}\right), \quad r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{c(t)}}(o_{i,t} \mid q, o_{i,<t})}. \tag{8}$$

我们在 DAPO 数据集上训练 DeepSeek-R1-Distill-Qwen-1.5B 进行 150 步，以评估我们提出的分块展开策略。实验在 64 个 A800 GPU 上进行，使用的批大小为 64，学习率为 3×10^{-6}。训练期间，我们将每个样本的展开次数设为 8。对于评估，我们报告了 16 次独立运行的平均性能。结果在表格 **??** 中展示。"Naive" 指展开阶段为每个查询生成完整响应；"Chunk-nk" 表示我们的分块展开策略，该策略在每次展开期间为每个输入生成最多 nk 个标记。指标"step" 和"sampling" 分别表示每次训练步骤的平均时间和每次步骤采样轨迹所花费的平均时间。所有时间数据都相对于天真动态采样基线进行了归一化。

从实验结果中，我们可以观察到，分块展开策略能够有效减少每步的训练和采样时间，同时保持性能。随着块大小的减小，每步的采样时间稳步下降，这证实了该策略在采样过程中缓解了因长轨迹引起的“气泡”现象，并提高了 GPU 的利用率。然而，当块大小从 8k 减少到 4k 时，虽然每步的采样时间明显减少，但每步的总训练时间基本未变。这是因为较小的块大小虽然缓解了采样瓶颈，但引入了更频繁的块级重要性采样的对数概率计算。结果，整体训练效率几乎没有提高。在未来的工作中，我们将进一步优化这种权衡，以在采样速度和对数概率计算之间实现更好的平衡。

1.3.6 实现细节

我们将训练批次大小设置为 256，微批次大小设置为 128。在整个训练过程中使用一个恒定的学习率 $1e-5$。为了适应 MiniCPM4 的架构设计，我们采用 μP 学习率策略。不像最近的方法那样去掉 KL 惩罚并在策略损失中引入熵约束，我们保留了系数为 0.001 的 KL 惩罚，并去掉熵约束以确保稳定的训练。最大响应长度设置为 32,768 个标记，这使模型能够执行扩展的思维链推理。在回滚阶段，温度和 top-p 都设置为 1.0，以鼓励更广泛的探索。对于每个查询，我们生成 16 个回滚以促进多样性和稳健性。

1.4 BitCPM4：三元大型语言模型的量化感知训练

由于 LLMs 具有较高的计算和内存要求，因此部署它们具有挑战性。模型量化通过降低参数精度来解决这个挑战，从而实现资源消耗更少的高效推理。极低比特量化（例如，1 位，2 位）最近引起了显著关注并显示出巨大前景（Wang et al., 2023; Ma et al., 2024; Xu et al.）。然而，为了构建这些极低比特 LLMs，PTQ 方法（Frantar et al., 2023）可能不足以保持模型性能，因此需要更有效的 QAT 方法（Liu et al., 2024b）。一些最近的努力，如 BitNet（Ma et al., 2025），甚至从头开始训练极低比特 LLMs。本文介绍了一种高效的 QAT 方法来构建一个有效的三元模型 BitCPM4，并展示了将高精度 LLMs 适配为极低比特版本的可行性。这意味着我们可以大大减小 QAT 过程中量化和反量化造成的额外开销。

1.4.1 高效量化感知训练

对于当前流行的量化方法，权重和激活通常都被量化。我们的初步实验表明，对于极低比特量化的 LLMs，量化激活增加了 QAT 的开销，而没有显著减少终端设备上的推理成本。因此，我们对模型权重而不是激活应用三值量化。目前最先进的三值 LLM，BitNet-2B (Ma et al., 2025)，从头开始使用 QAT 对 4T tokens 进行训练，并达到了令人印象深刻的性能。相比之下，我们的策略是用经过预训练的高精度检查点初始化三值模型，以减少 QAT 所需的训练 tokens。

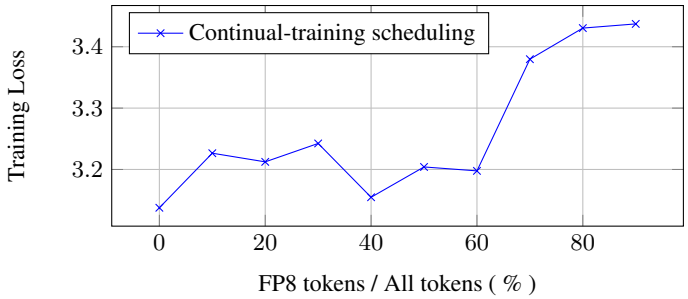

Figure 4: 语言建模损失与 QAT 训练后代币比例（全稳定阶段代币的比例）之间的关系。

为了验证从高精度检查点初始化的模型应用 QAT 的可行性，我们首先在我们的 ModelTunnel 中对一个具有 5M 参数的模型进行了初步实验，使用了总共 2B（400N）个 token。在这些实验中，我们采用了全 WSD 学习率调度器，其中热身阶段和稳定阶段占总 token 的 80 %，衰减阶段占剩余的 20 %。这些初步实验包括两个训练阶段：首先训练一个 FP8 模型，然后通过 QAT 将该 FP8 模型转换为三元模型。在我们的 ModelTunnel 中进行的充分实验表明，在第二阶段开始时重新热身学习率对于维持 FP8 模型的性能至关重要。为此，我们在第一阶段采用了 1e-2 的学习率，而在第二阶段，我们使用了 5e-3 的学习率。在保持总训练 token 数量不变的情况下，我们调整 FP8 训练阶段和 QAT 阶段之间的分配比例，并记录每种配置的最终收敛损失。

如图 4 所示，当用于 QAT 的 token 比例超过总训练 token 的 40 % 时，即相当于学习率衰减阶段所用 token 数量的两倍时，最终损失接近于从头通过 QAT 训练三进制模型。此外，我们在一个 150M 参数的模型上进行验证实验，结果证实合理的持续训练调度可以达到与从头执行 QAT 相同的效果以获得有效的三进制模型。为此，我们仅使用学习率衰减阶段所用 token 数量的两倍构建 BitCPM4。

1.4.2 极低比特大型语言模型的讨论

我们最终训练了两种规模的三值模型：一个是基于 MiniCPM4 -0.5B 训练的 BitCPM4-0.5B，另一个是基于内部实验模型训练的 1B 参数模型。整个 QAT 过程使用了 350B 个标记。

我们将我们的模型与其他相关模型进行比较，结果如表 2 所示。在 0.5B 参数级别上，BitCPM4-0.5B 在知识相关任务（MMLU、CMMLU、C-EVAL 等）上优于 qwen3-0.6B。在 1B 参数级别上，BitCPM4-1B 的表现与竞争的 2B 参数模型相似。由于 BitCPM4 所需的 tokens 数量仅为 BitNet-2B 的 10 %，这意味着从头实现 QAT 是没有必要的，进一步表明我们提出的 QAT 方法可以在较低的训练成本下提供有竞争力的结果。

然而，我们的 0.5B 参数模型在更具挑战性的数学和编码任务上表现相对较弱，我们将其归因于较小的模型尺寸限制了推理能力。现有的量化工作表明，量化效果遵循与模型大小有关的缩放定律（Ouyang et al., 2024; Kumar et al., 2024）。基于这一规律，我们计划在未来的工作中将 QAT 方法应用于更大的模型。此外，针对极低位模型的运算符也是一个需要充分考虑的问题，这也将是我们未来的工作。

2 高效推理与部署

由于终端设备（如移动设备和个人计算机）在计算、存储容量和功耗方面有严格的限制，如何在有限的硬件资源下实现对大型语言模型（LLMs）的高效推理已成为一个关键的技术挑战。在本节中，我们将介绍我们的推理系统 CPM.cu 和部署系统 ArkInfer。

2.1 CPM.cu：轻量级且高效的 CUDA 推理框架

我们首先开发了一个专为终端 NVIDIA 芯片优化的轻量级推理框架。除了基本功能如静态内存管理和内核融合之外，我们还实现了高效的猜测采样，并为 InfLLM v2 集成了高效的稀疏注意力内核。猜测采样是一种加速 LLM 推理的关键技术，特别是在资源有限的终端设备中。该方法采用了一个“草稿-验证”范式，其中轻量级的草稿模型生成候选标记序列，然后由目标 LLM 并行验证。最近在

Table 2: BitCPM4 与其他代表性模型的比较。带星号的数据来自该模型的原始论文，其余数据由我们自行复现。

Model	Qwen3	Llama3.2	Gemma3	BitNet	BitCPM4	BitCPM4
# Parameter	0.6B	1B	1B	2B	0.5B	1B
Precision	BF16	BF16	BF16	Ternary	Ternary	Ternary
MMLU	42.95	46.89	41.64	53.17*	49.88	59.24
CMMLU	42.05	23.73	25.09	27.61	55.88	68.84
CEval	45.53	36.74	31.83	29.36	57.51	69.06
BBH	28.32	25.42	33.21	49.83	43.13	57.64
GSM8K	61.71	39.76	61.26	58.63*	25.55	60.80
MATH500	50.20	17.20	43.20	42.40	10.20	34.00
MBPP	47.86	47.47	59.92	47.08	46.69	61.48
HumanEval	40.85	40.85	42.07	38.40	29.88	37.20
Average	44.93	34.76	42.28	43.31	39.84	56.03

猜测采样方面的进展，如 EAGLE-2，采用了树状草稿过程。通过设计适用于基于树的猜测采样的高效注意力内核和实现融合验证内核，我们实现了优化的猜测采样速度。我们还在框架内实现了一个高效的 InfLLM v2 稀疏注意力内核。

基于该框架，我们识别出面向终端模型的推测采样效率瓶颈在于草稿模型的语言建模头。为了解决这个问题，我们提出了 FR-Spec (Zhao et al., 2025)，该方法根据标记频率修剪草稿模型的词汇，同时保留目标模型的完整词汇以维持其生成质量。我们进一步探索将推测采样与 4 位量化的 GPTQ (Frantar et al., 2023) 模型结合。为了保持模型性能，我们首先探索了一种改进的量化方案，P-GPTQ，随后验证了在 SpecMQuant (Zhang et al., 2025b) 中将推测采样与量化结合以及与长上下文处理集成的可行性。

2.1.1 基于频率排序的词汇构建和草稿验证

投机采样的有效性在很大程度上依赖于起草和验证阶段的效率。最近的进展如 EAGLE-2 (Li et al., 2024c) 已通过使用单层 Transformer 进行起草，显著减少了起草的开销，从而取得了显著成效。当代模型趋向于采用更大的词汇量以提高分词效率，这在语言建模头中引入了显著的计算开销，即使只有单层，使得起草过程变慢。我们的研究表明，从小词汇量模型过渡到大词汇量模型会大幅增加起草时间，形成新的性能瓶颈，限制了投机采样技术的有效性。

为了解决这一挑战，我们引入了 FR-Spec (Zhao et al., 2025)，这是一种频率排序的预测抽样框架，通过战略性压缩词汇空间来优化草稿候选项的选择。我们的方法利用了自然语言中已广为人知的词元频率的长尾分布，其中一小部分高频词元占据了大多数出现次数。通过将草稿搜索限制在一个频率优先的词元子集，FR-Spec 在保持验证过程的数学等价性和最终输出分布正确性的同时，将语言建模头的计算开销减少了多达 75% 。

FR-Spec 引入了一种频率排序的投机采样方法，该方法在优化草稿阶段的同时，保持验证过程的数学等效性。如图 5 所示，该框架基于词汇空间压缩的原理运行，其中草稿模型的计算范围被限制在一部分高频标记，从而在减少开销的同时保持可接受的草稿质量。

基于频率的词汇子集构造。FR-Spec 的基础在于系统地识别和选择高频词元。我们对大规模预训练数据进行语料库层级的分析，以建立全面的词元频率排名。设 $f(t)$ 表示词元 t 在语料库 $\mathcal{C}$ 中的频率。我们按频率降序排列所有词元 $t \in \mathcal{V}$，并选择排名前 k 的词元以形成我们的简化词汇子集：$\mathcal{V}_{\text{high}} = \{t_1, t_2, \ldots, t_k\}$ where $f(t_1) \geq f(t_2) \geq \cdots \geq f(t_k)$。我们的实证分析表明，选择词汇表的大约 25 % （$k = 0.25 \times |\mathcal{V}|$ ）能够提供最佳的性能，在显著减少计算量的同时捕获 95 % 的词元出现次数。

修改后的起草计算。在标准的投机采样中，草稿模型计算整个词汇表 $\mathcal{V}$ 上的概率分布。FR-Spec 通过将计算限制在缩减后的词汇子集 $\mathcal{V}_{\text{high}}$ 上来修改此过程。给定原始语言建模头矩阵 $\boldsymbol{W}_{\text{LM}} \in \mathbb{R}^{|\mathcal{V}| \times d}$，我们通过提取对应于高频词元的行来构建缩减矩阵 $\tilde{\boldsymbol{W}}_{\text{LM}} \in \mathbb{R}^{|\mathcal{V}_{\text{high}}| \times d}$：

$$\tilde{\boldsymbol{W}}_{\text{LM}}[i,:] = \boldsymbol{W}_{\text{LM}}[\mathcal{V}_{\text{high}}[i],:], \quad i = 1, \ldots, |\mathcal{V}_{\text{high}}|. \tag{9}$$

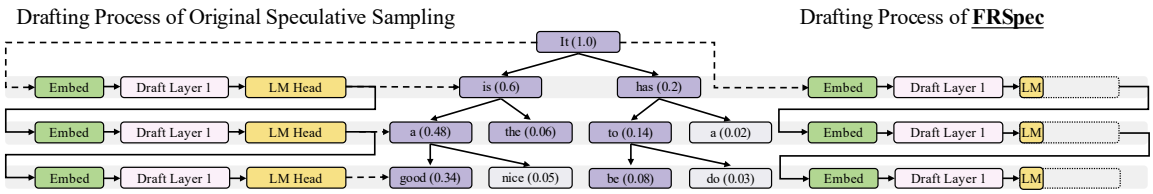

Figure 5: FR-Spec 的说明，它要求草案模型使用一个减少的词汇子集。

修改后的起草计算变为：

$$\mathcal{D}_{\text{FR}}(\boldsymbol{x}) = \text{Softmax}(\boldsymbol{H}_{\text{draft}}(\boldsymbol{x})\tilde{\boldsymbol{W}}_{\text{LM}}^{T}), \tag{10}$$

其中 $\boldsymbol{H}_{\text{draft}}(\boldsymbol{x}) \in \mathbb{R}^{n\times d}$ 代表了草稿模型对输入序列 $\boldsymbol{x}$ 的隐藏状态。

验证过程。FR-Spec 的一个关键设计原则是保持验证过程的数学正确性和分布等价性。目标 LLM 继续在完整的词汇空间 $\mathcal{V}$ 上运行，确保最终输出分布与标准的推测采样方法相同：$\mathcal{P}_{\text{target}}(\boldsymbol{x}) = \text{Softmax}(\boldsymbol{H}_{\text{target}}(\boldsymbol{x})\boldsymbol{W}_{\text{LM}}^{T})$。这种保留确保了 FR-Spec 在实现计算加速的同时产生统计等价的结果。

计算复杂性分析。FR-Spec 的计算优势显著且定义明确。语言建模头的计算复杂性从 $O(nd|\mathcal{V}|)$ 降低到 $O(nd|\mathcal{V}_{\text{high}}|)$，其中 n 是草稿序列长度，d 是隐藏维度。减少因子为 $\frac{|\mathcal{V}|}{|\mathcal{V}_{\text{high}}|}$。同样，随着输入维度从 $\mathbb{R}^{n\times|\mathcal{V}|}$ 减少到 $\mathbb{R}^{n\times|\mathcal{V}_{\text{high}}|}$，softmax 的计算复杂性也相应减少。对于典型的配置，其中 $|\mathcal{V}_{\text{high}}| = 0.25\times|\mathcal{V}|$，这表示草稿模型的语言建模头的计算开销降低了 $4\times$。

FR-Spec 被设计为一种即插即用的增强技术，可以无缝整合到现有的推测采样技术中，而无需对模型进行重新训练或结构修改。该方法可以应用于各种推测采样框架，只需用频率排序的变体取代标准的起草计算即可。

2.1.2 P-GPTQ：适用于终端设备的前缀感知后训练量化

正如我们上面提到的，PTQ 和 QAT 都是模型量化的重要方法。与 QAT 相比，PTQ 更轻量且更易于实施。用于设备上的 PTQ 需要同时对权重和激活进行量化，以适应受限的计算资源。

最近的研究表明，大多数大型语言模型在初始标记位置上表现出大规模激活 (Sun et al., 2024b)，显著降低了激活量化的保真度。为了解决这一挑战，我们遵循 PrefixQuant 方法 (Chen et al., 2024) 来隔离这些初始标记激活异常值。具体来说，我们的解决方案是在预处理过程中存储初始键值激活，从而防止在前几个输入标记的前向传播过程中出现过大的激活幅度。

尽管 PrefixQuant 在某种程度上解决了推理期间的激活异常问题，但我们观察到初始标记偏差也显著影响了权重量化校准过程。基于这一见解，我们开发了前缀感知的 GPTQ (P-GPTQ)，这是 GPTQ 方法的扩展 (Frantar et al., 2023)，能够在 Hessian 计算过程中消除初始标记干扰。形式上，GPTQ 从校准数据 $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ 计算 Hessian 矩阵 $\boldsymbol{H}$，如下所示

$$\boldsymbol{H} = \boldsymbol{X}^{\top}\boldsymbol{X}. \tag{11}$$

我们的实验分析表明，在计算用于向下投影层的协方差矩阵 $\boldsymbol{H}$ 时，尤其是在更深的 Transformer 块中，句子开头的标记和一些初始标记始终引入显著的统计偏差。这些初始位置的激活量显示出比后续标记大 10 倍的幅度，不成比例地主导了协方差结构，并导致次优量化参数。为了减轻这种偏差，P-GPTQ 实施了一种位置感知校准策略。通过对多个层级的实证分析，我们发现标记位置从 $s = 4$ 开始表现出稳定的统计特征。我们仅在计算稳定标记位置时考虑 Hessian 矩阵

$$\hat{\boldsymbol{H}} = \boldsymbol{X}_{\text{valid}}^{\top}\boldsymbol{X}_{\text{valid}}, \tag{12}$$

，其中 $\boldsymbol{X}_{\text{valid}} = \boldsymbol{X}_{[s:]}$ 排除了前 s 个位置。P-GPTQ 的理念保持了与其他量化技术的兼容性，包括 Quarot (Ashkboos et al., 2024) 这样的旋转方法和 AWQ (Lin et al., 2024b) 这样的平滑方法，能够无缝集成到现有的量化管道中。

我们在将所有线性层量化为每组 INT4 格式的设置下评估 P-GPTQ 及其扩展。校准使用了从训练数据集中随机选择的 1,024 个序列。表 3 展示了各量化方法的比较结果，其中 S 表示通过 AWQ 预处理应用的平滑过程。结果表明，S-P-GPTQ 在量化方法中实现了优越的性能，与 FP16 基线相比，表现出最小的性能下降。

Table 3: 不同量化方法的评估结果。

Benchmark	FP16	GPTQ	P-GPTQ	S-GPTQ	S-P-GPTQ
MMLU	75.55	75.01	75.21	75.05	75.40
CMMLU	82.12	81.36	81.36	81.39	81.95
CEval	81.42	80.08	80.92	80.71	80.70
BBH	70.70	69.78	69.97	70.17	70.17
GSM8K	82.41	80.71	80.53	79.83	80.67
Math500	60.20	59.72	59.62	59.80	60.00
MBPP	76.65	73.54	75.68	75.49	75.49
Average	75.58	74.31	74.76	74.63	74.91

2.1.3 推测采样遇上量化和长上下文

目标模型 的量化在 SpecMQuant (Zhang et al., 2025b) 中，我们提供了一个系统的分析，用于在将投机取样应用于量化模型时需要考虑的关键因素。例如，当使用具有 W4A16 目标模型的 EAGLE-2（例如，通过 GPTQ (Frantar et al., 2023) 量化的模型）时，由于量化模型缓解了目标模型的内存访问瓶颈，起草过程应该使用比非量化目标模型更少的草稿标记。具体来说，设 n 个草稿标记的验证时间为 $T_v(n)$，目标模型的普通解码时间为 T_t，在量化模型中，验证到解码时间的比率 $T_v(n)/T_t$ 随着 n 的增加而显著更快地增长，与非量化模型相比。这导致使用更多草稿标记所带来的接受长度增加被验证时间的大幅延长所抵消。

的草稿模型量化。我们进一步对 EAGLE-2 草稿模型进行量化。这可能使起草过程更加快速。此外，将草稿模型压缩到 4 位版本也使其适合于内存受限的终端部署。然而，QSpec (Zhao et al., 2024) 发现对 EAGLE-2 使用 GPTQ 会显著降低接受率。因此，我们改为在 EAGLE-2 上使用 QAT（第 1.4 节）。我们验证了，通过我们的 QAT 方法量化的 EAGLE-2 对推测采样过程的平均接受长度没有产生负面影响。

面向目标模型 的 InfLLM v2 我们为长上下文场景实现了 InfLLM v2 稀疏注意力内核。为了支持在推测采样中进行树形草稿验证，假设草拟的标记总数为 n，我们仅为最后的 $n \times n$ 区域构建一个局部的 2d 注意力掩码，并在将其传递给 InfLLM v2 内核之前通过 uint64 位打包来压缩该掩码。

滑动窗口用于草稿模型 在长上下文场景中，如果在预测性采样中草稿模型使用完整注意力，它将显著增加模型的首个标记延迟。根据 TriForce (Sun et al., 2024a) 中的方法，我们对草稿模型应用滑动窗口注意力。我们的实验表明，这不仅最小化对首个标记延迟的影响，还提高了草稿的准确性。

2.2 ArkInfer：跨平台部署系统

除受限的计算资源外，终端侧芯片的碎片化构成了另一个显著的障碍。这种碎片化需要为每次新模型发布调整模型以适应多个平台和芯片类型，从而导致复杂的调整和部署。这使得工程工作量巨大，几乎不可能让模型在所有平台上高效运行。

这个问题的核心在于解耦和高效代码重用：如何能够自动地将单一的技术开发和工程工作应用于多个平台？

为了解决这些痛点，我们提出了 ArkInfer，一种新颖的跨平台部署系统。ArkInfer 旨在通过提供高效的推理速度和作为各种模型应用的多功能跨平台兼容层，来克服终端芯片的碎片化问题。为此，我们引入了三个关键解决方案：(1) 跨平台兼容的架构设计，(2) 可重用且高效的推测和约束解码方案，以及 (3) 可扩展的模型库前端。

2.2.1 跨平台兼容架构设计

ArkInfer 的架构设计基本上是由在分散的终端硬件环境中实现统一、高效部署的需求所驱动的。支持多种平台，例如联发科、Nvidia、高通和瑞芯微，它们各自拥有自己的原生推理框架（如用于 CPU 的 NeuroPilot、Genie、RK-LLM、TensorRT-LLM 和 llama.cpp），ArkInfer 可无缝地将这些框架整合为可调整的后端。

ArkInfer 的核心实现了一个强大的抽象层。该层具有一个适配器系统，可以对不同后端的多样化 API 进行标准化，向更高层组件呈现一致的接口。这确保了无论底层硬件或框架如何，都能实现无缝交互。数据处理通过一个统一的 Tensor 结构得到进一步简化，该结构封装了多种数据类型和维度，以在整个系统中进行一致的操作。为提高 LLM 效率，一个专用的 KV 缓存管理器智能地协调历史状态的存储和检索，优化后续的令牌生成。

这个架构的核心组件是一个抽象的执行器接口，它管理所有与模型相关的过程的运行时执行，输入和输出由基本张量类型定义。这包括核心神经网络的执行（处理各种输入模态如文本、图像和音频的编码，以及自回归解码）、复杂的采样技术生成多样化的输出、以及全面的预处理能力以准备数据供模型输入。除此之外，ArkInfer 还通过组合这些基本执行器来组织复杂的预训练模型，并管理与外部工具调用功能的交互。

这种设计在执行器粒度上实现了异构调度，使我们能够充分利用多样化的计算资源。此外，通过跟踪执行器的执行，我们可以追踪数据和操作的流动，这极大地促进了调试和性能分析，特别是在每个阶段的精度对齐——这通常是端侧适配中的一个常见痛点。

2.2.2 可重用且高效的推测和约束解码方案

高效的 LLM 推理技术通常分为三类：量化、稀疏性和自回归过程的加速。虽然前两者，例如 GPTQ、MoE 以及我们的 InfLLMv2，通常与特定硬件或操作符实现密切结合，但像推测采样和约束解码这样的加速技术与底层硬件的耦合相对较松。这种解耦使得我们可以在一个部署框架中实现这些优化，并在多个芯片架构上启用它们。因此，ArkInfer 集成了推测采样和约束解码功能。我们的设计理念侧重于在现有执行后端内部实现普遍适用性和易于集成。

ArkInfer 生成输出标记的能力的核心在于一个核心组件，它负责处理输入并协调自回归生成过程。该组件支持多种高级解码策略，以满足多样的推理需求：

加速推测解码 为了提高推理速度，除了上述的推测采样方法外，ArkInfer 还结合了一种基于 BiTA 算法（Lin et al., 2024a）的先进推测解码机制。该技术是一种策略性选择，因为它在不需要额外草稿模型或专门的架构变化的情况下显著提升性能，从而简化了在资源受限的终端设备上的部署，同时保持了高输出质量。

约束解码 为确保输出符合特定格式，例如 JSON 或 SQL，ArkInfer 使用强大的约束解码方法，利用 Guidance 算法。选择这种方法是因为它在强制结构符合性和提供确定性响应方面有卓越的能力，这对于需要结构化或精确输出的应用程序至关重要。

2.2.3 可扩展模型库前端

在边缘设备上部署模型的一个关键障碍来自于模型文件结构的碎片化。不同的芯片制造商经常提出自己的特定要求和格式，导致了复杂且低效的部署流程。我们认为，最佳的方法是维护一个集中化的“模型动物园”，提供广泛选择的预适应模型。

为了解决这个问题，我们设计了一个可扩展的、多平台的 ArkInfer 前端。这一接口让用户可以直接访问并执行我们模型库中的各种模型，从而显著简化了 MiniCPM 和其他模型在各种设备上的部署。除了加快我们模型库的增长和维护，我们还创建了一个自动化的模型转换管道。这个系统可以高效地将模型转换为不同平台所需的格式，大大加速了我们模型库的持续开发。

基于高效的训练和推理机制，我们构建了 MiniCPM4 -8B 和 MiniCPM4 -0.5B。在本节中，我们在几个开源基准上评估我们的模型的有效性和效率。

2.3 实验设置

基准 MiniCPM4 主要在中文和英文语料库上进行了预训练。因此，我们选择以下数据集来评估我们的模型，包括知识密集型评估集 MMLU（Hendrycks et al., 2020）、CMMLU（Li et al., 2024a）和用于英语和中文的 CEval（Huang et al., 2023），以及推理评估集，包括一般推理的 BigBench Hard（BBH）（Suzgun et al., 2023），数学推理 GSM8K（Cobbe et al., 2021）和 MATH500（Hendrycks et al., 2021），以及代码推理 MBPP（Austin et al., 2021）和 HumanEval（Chen et al., 2021）。我们采用 OpenCompass（Contributors, 2023）作为我们的评估框架。

Table 4: MiniCPM4 及其他开源 LLMs 的评估结果。

Models	Qwen3	Llama3.2	Gemma3	MiniCPM4	Qwen3	GLM4	Gemma3	LLaMA3.1	Phi4	MiniCPM4
# Parameter	0.6B	1B	1B	0.5B	8B	9B	12B	8B	14B	8B
# Train Data	36T	9T	2T	1T	36T	10T	12T	15T	10T	8T
MMLU	42.95	46.89	41.64	55.55	77.55	75.90	73.36	69.38	81.61	75.83
CMMLU	42.05	23.73	25.09	65.22	77.58	74.49	62.52	54.41	67.56	80.62
CEval	45.53	36.74	31.83	66.11	80.35	74.09	62.23	52.66	64.28	81.36
BBH	28.32	25.42	33.21	49.87	69.43	61.36	66.66	44.34	72.79	76.73
GSM8K	61.71	39.76	61.26	52.08	93.25	89.39	94.16	84.08	94.77	91.51
MATH500	50.20	17.20	43.20	29.60	83.20	66.00	82.20	48.20	79.60	78.60
MBPP	47.86	47.47	59.92	59.14	77.04	74.71	84.44	68.09	80.54	78.99
HumanEval	40.85	40.85	42.07	46.34	85.98	82.32	83.54	70.73	86.59	85.37
Average	44.93	34.76	42.28	52.99	80.55	74.78	76.14	61.49	78.47	81.13

基线模型　我们将 MiniCPM4 -8B 和 MiniCPM4 -0.5B 与几个广泛采用的开源大语言模型进行比较。具体来说，对于 MiniCPM4 -0.5B，我们选择了几个大约有 10 亿参数的模型，包括 Qwen3-0.5B (Yang et al., 2025) 、Llama3.2-1B (Dubey et al., 2024) 、Gemma3-1B (Team et al., 2025) 。这些模型经过了数万亿令牌的充分训练，并使用了知识蒸馏技术，从而保证了它们的有效性。对于 MiniCPM4 -8B，我们选择了大约有 100 亿参数的模型作为基线，包括 Qwen3-8B (Yang et al., 2025) 、GLM4 (0414 版本) (GLM et al., 2024) 、Gemma3-12B (Team et al., 2025) 和 Phi4-14B (Abdin et al., 2024) 。

预训练流程　我们采用 μ P 作为我们的基础模型架构。在模型预训练之前，我们首先搜索包含数百万参数的模型的超参数，包括学习率、批量大小和参数初始设置。然后，我们遵循一个四阶段的流程来预训练 MiniCPM4 。首先，我们使用 7 万亿个标记进行稳定的预训练阶段，学习率为 $7 \times 10^{-}3$ 。接着，我们进行一个退火预训练阶段，使用 1 万亿个标记。在这两个阶段中，上下文长度设置为 4 K。为了实现长序列处理，我们将上下文窗口从 4 K 扩展到 $32K$ 。在此阶段，我们用 20 亿个标记训练模型，并使用 LongRoPE (Ding et al., 2024) 作为我们的位置信编码。值得注意的是，尽管我们仅在 32 K 上下文中训练模型，MiniCPM4 可以处理使用 YaRN (Peng et al., 2023) 的 128 K 序列。在三个预训练阶段之后，我们进行监督微调和强化学习，以使我们的模型有用、诚实和无害。为了实现最终的推理加速，我们进一步对推测头进行额外训练，以提高草稿模型的接受长度。值得注意的是，基于 MiniCPM4 ，我们将在未来几周发布端侧推理模型。

我们在表格 4 中展示了 MiniCPM4 和基线模型的评估结果。从结果中可以观察到：1）我们的两个模型在相似规模的模型中都达到了最先进的性能，证明了我们训练策略的有效性。此外，我们的模型在参数量显著更多的多个开源大语言模型上表现更佳。例如，MiniCPM4 -0.5B 相比于 Llama3.2-1B 和 Gemma3-1B，尽管这些模型的参数规模是 MiniCPM4 的两倍，仍然表现出色。同样地，MiniCPM4 -8B 超越了 Gemma3-12B 和 Phi4-14B。这进一步验证了通过利用高质量数据和高效学习算法，MiniCPM4 可达到卓越的性能。2）与这些开源模型相比，MiniCPM4 以显著较低的训练成本达到了优秀的性能。具体来说，MiniCPM4 的性能与 Qwen3 相当，而 Qwen3 使用了 36 万亿个 token 进行训练，而 MiniCPM4 仅使用了 8 万亿个 token ——仅为 Qwen3 训练数据规模的 22 % 。3）在基线模型中，包括 Qwen3-0.6B/8B、Llama3.2-1B 和 Gemma3-1B/12B，采用了知识蒸馏训练策略，使用更大的教师模型指导终端侧模型的训练。我们的实验结果显示，尽管只用真实值作为监督信号，我们的模型仍表现出强劲的性能。蒸馏过程需要大量计算资源来部署教师模型。未来，我们将探索更高效的模型蒸馏策略，以进一步提升终端侧模型的性能。

在 MiniCPM4 中，我们使用稀疏注意力机制将上下文窗口扩展到 32 K。在本段中，我们在长序列理解任务上评估 MiniCPM4 。具体而言，我们遵循 Hsieh et al. (2024) ，并在大海捞针任务 (RULER-NIAH) 上评估我们的模型。我们应用 YaRN (Peng et al., 2023) 将 MiniCPM4 的上下文窗口扩展到 128 K，并使用 128 K NIAH 评估 MiniCPM4 。

结果显示在图 **??** 中。从结果中我们可以观察到：1）MiniCPM4 可以在长序列上达到令人满意的性能，并在大海捞针任务中实现 100% 的准确度。而且对于每个 token，MiniCPM4 只需要模型关注 6 K 个上下文 token，这意味着在 128 K 上下文中，MiniCPM4 的稀疏度仅为 5% 。2）MiniCPM4 在上下文窗口外推方面表现良好。即使我们只在 32 K 上下文上对模型进行预训练，MiniCPM4 也可以在 4× 上下文长度上实现 100 个% 的准确度。在后续部分中，我们将 MiniCPM4 应用于调查生成任务，

该任务要求模型读取和撰写长文档。而且 MiniCPM4 比其他基线模型表现更好，表明了 MiniCPM4 在长序列处理中的有效性。

为实现终极的推理加速，我们在 MiniCPM4 中构建了一种稀疏注意力机制 InfLLM v2，采用 FR-Spec 猜测采样算法，提出了前缀感知量化算法，并构建了我们专门的推理框架，以在终端设备上实现最大速度提升。为了验证我们提出算法的有效性，我们在本节对两个典型的终端芯片测试了我们模型的效率。具体来说，我们选择了两个边缘芯片：Jetson AGX Orin 和 RTX 4090。前者广泛应用于汽车芯片和机器人等终端场景，而后者主要用作个人计算机中的计算设备。

评估结果如图 1 所示。我们评估了 Llama3-8B（Dubey et al., 2024）、GLM4-9B（GLM et al., 2024）、Qwen3-8B（Yang et al., 2025）和 MiniCPM4 在 32 K 到 128 K 范围内序列的吞吐速度。从结果中可以观察到：1）与具有相似参数大小的开源 LLMs 相比，我们在预填充和解码场景中均能实现一致的加速。特别是，与 Qwen3-8B 相比，我们在 Jetson AGX Orin 上实现了大约 7 倍的解码加速。结果证明了我们方法的有效性。2）随着文本长度的增加，我们模型的效率优势变得更加明显。这是因为稀疏注意机制能够有效减少处理长文本的计算和内存访问开销。随着模型需要处理的文本长度逐渐增加，传统密集注意机制的内存访问开销快速增长，而 InfLLM v2 需要访问的上下文块数量保持不变，只有语义内核的表示随序列长度缓慢增长。因此，在长序列处理中，MiniCPM4 可以持续高效地处理长文本。

3 应用

的效率在多种场景中解锁了引人注目的能力。我们突出三个关键应用：（1）可信问卷生成展示了其在长序列处理中的强大功能，这对于理解和综合大量文档中的复杂信息以生成准确的摘要至关重要。（2）使用模型上下文协议的工具使用对于以代理为中心的部署是关键，允许 MiniCPM4 可靠地解释指令、上下文和状态信息，以无缝地与外部工具和 API 进行交互——这是构建健壮、能力强的代理的基础。

3.1 MiniCPM4 - 调查：可信的调查生成

即使是经验丰富的研究人员，进行全面的文献调查也面临着显著的挑战。这一工作需要高级能力，包括收集相关资源、整合和综合多样信息、找出关键挑战以及预测未来研究方向。幸运的是，由于 LLM 推动的 AI 技术的快速发展，这使得自动化深度研究变得越来越可行。最近的努力（Wang et al., 2024c; Li et al., 2025b; Wang et al., 2025a），例如像 OpenAI Deep Research [5] 和 Gemini Deep Research [6] 等项目，利用现代 LLM 的长篇推理能力构建实用的调查系统，简化人类研究人员掌握广泛学术文献和快速理解新研究领域的过程。

设备端调查系统提供了重要的优势，补充了基于云的服务。关键是，对于那些有严格保密需求且无法将资源上传到云端的用户而言，本地部署的内部模型是唯一可行的选择。这保证了数据的完整性和控制。此外，在设备端部署小规模模型在推理时显著降低计算成本，这在调查撰写过程中固有的高 token 消耗下是一个关键考虑因素。这使得本地化系统对于许多应用而言既安全又经济合理。

为此，我们提出了 MiniCPM4-Survey，这是一种构建在 MiniCPM4 -8B 基础上的模型，能够生成可信的长篇综述论文，同时相较于显著更大的模型保持有竞争力的性能。具体来说，我们的模型以计划-检索-写作的方式进行工作，其中包括三个核心阶段：（1）计划：从全局视角定义综述的整体结构，具体说明每个章节、小节和段落中要讨论的内容；（2）检索：根据规划者生成的大纲生成适当的检索关键词，并查询知识库以获取相关文献；以及（3）写作：综合检索的信息以生成连贯的章节级内容，循环进行直至完成整个综述。

为增强 MiniCPM4 -8B 在该任务上的能力，我们整理和处理了一大批由专家撰写的综述论文，构建一个高质量的训练数据集。同时，我们收集了一系列研究论文来建立一个检索数据库。我们进一步引入一个多阶段的训练流程，包括：有监督的微调、章节级别的强化学习和综述级别的强化学习。通过结合紧凑模型的高效性与严格的训练方法论，MiniCPM4-Survey 的性能可与大规模的 LLMs 相媲美，有时甚至超越，但仍保持良好的可访问性和成本效率。

[5] https://openai.com/index/introducing-deep-research

[6] https://gemini.google/overview/deep-research

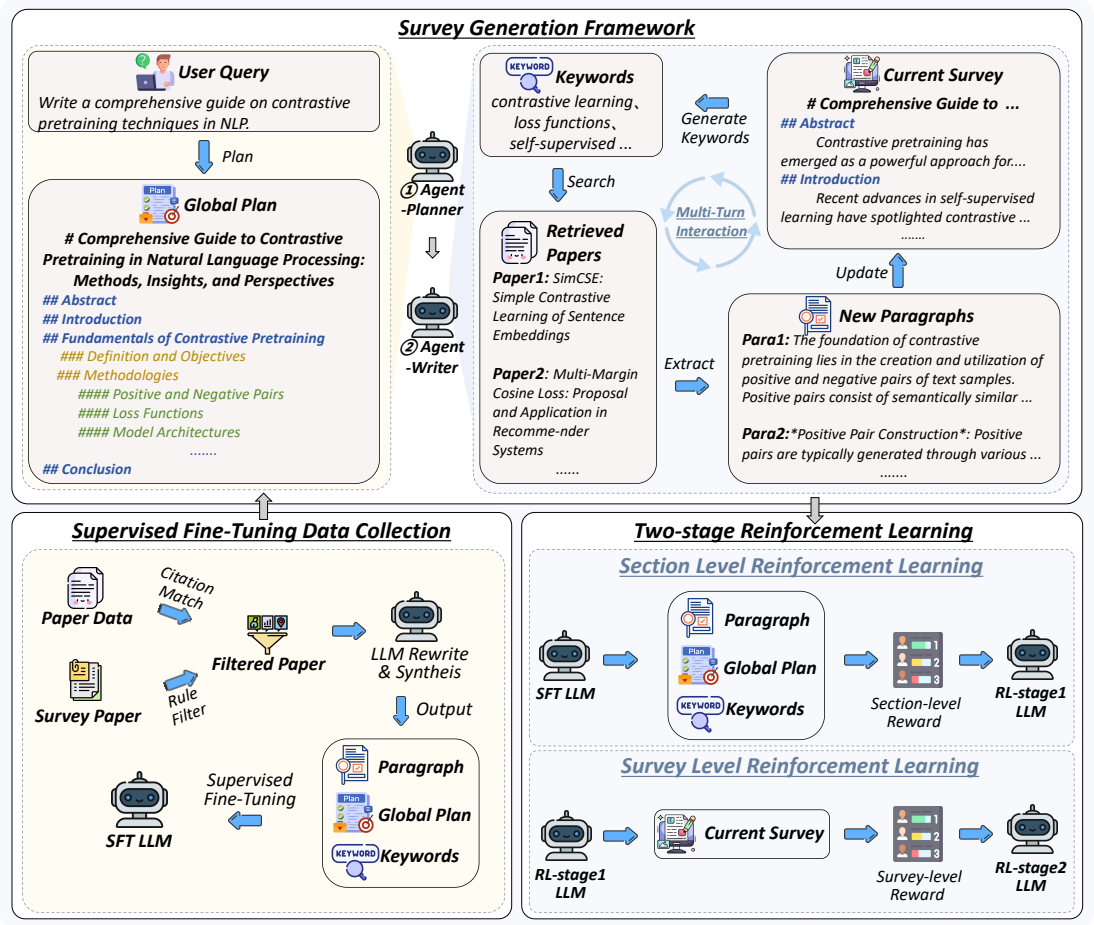

Figure 6: MiniCPM4-Survey 的概述。

3.1.1 数据构建

如前所述，调查生成流程涉及几个关键阶段，包括规划、迭代检索和内容生成，这些阶段的全面覆盖需要训练数据。为了提高 MiniCPM4 的调查生成能力，我们优先构建主要源自学术调查的高质量数据集，确保稳健的训练效果。具体而言，我们从 Kaggle [7] 收集了大约 271 万篇论文摘要作为基础数据。随后，我们结合使用 Faiss [8] 构建了一个高效的检索索引。

为了确保数据的可用性和质量，在构建训练数据集之前，原始数据经过了严格的预处理步骤。这些步骤包括过滤掉多模态元素，如表格和图像，移除数据库中未索引的参考文献，以及标准化数据格式。需要注意的是，除了简单的基于规则的方法外，我们还大量使用了大型语言模型（LLMs）来重写和优化原始数据，从而提高其一致性和相关性。此外，遵循 Wang et al. (2025a) 提出的调查生成框架，我们从预处理后的数据中综合提取了三种不同类型的数据——查询、计划和检索关键词。这些综合数据集根据 Query2Plan 和 Plan2Survey 阶段系统地构建，分别得到 3,750 个和 61,684 个训练样本。

3.1.2 训练策略

鉴于问卷生成对基础模型构成了相当的复杂性，我们首先采用了小规模的监督微调（SFT），使用有限的数据集来实现有效的冷启动性能，并提高积极样本抽样的可能性。随后，我们转向了强化学习（RL）阶段，以实现稳定和持续的性能提升。值得注意的是，我们将 RL 过程分解为两个不同的阶段：

[7] https://www.kaggle.com/api/v1/datasets/download/Cornell-University/arxiv
[8] https://github.com/facebookresearch/faiss

Table 5: 针对不同代理能力的奖励系统。

Agent Ability	Metrics	Judgement	Description
Planning	Structure Rationality	LLM & Rule	Whether the outline is reasonable.
	Structure Similarity	Rule	Whether similar to the golden plan.
	Truthfulness	LLM	Whether the content in the plan is real and reliable.
Searching	Recall Score	LLM	Whether the recalled papers include the golden papers.
Content Writing	Length	Rule	Whether the length of each section is reasonable.
	Language (en)	Rule	Whether the response is in English only.
	Relevance	LLM	Whether focus on the user's query.
	Coverage	LLM	Whether covers a wide range of related topics.
	Depth	LLM	Whether reflects deep and dynamic discourse.
	Novelty	LLM	Whether covers several new aspects of the user's query.
	Redundancy	LLM	Whether concise and no repeat sections.
Citation Writing	Hallucination	Rule	Whether all the citations appear in the retrieved information.
	Fact Score	LLM	Whether the fact claims are consistent with citations.

首先在章节层面优化奖励，以确保每个章节内的上下文一致性和结构精确性，然后将优化目标转向整体问卷级别的目标，增强整体的一致性、深度和主题相关性。这个逐步挑战的优化策略显著提高了模型的最终性能。

强调在调查生成能力的训练过程中遇到的三个主要挑战非常重要。首先，使用传统指标（如困惑度 PPL）难以可靠地评估生成调查的质量，因此需要创建一个专门的奖励系统。其次，调查生成涉及多个复杂阶段，要求强大的上下文管理，以确保保留重要信息，同时促进高效推理。第三，生成过程中固有的迭代检索和评估需要低延迟交互，以保持高效的强化学习训练周期。为了解决这些挑战，我们特别实施了以下优化：

奖励设计：意识到奖励设计在强化学习中起到的关键作用，我们建立了一个全面的奖励框架来评估关键模型能力，包括规划、检索、内容创建和引用准确性。该系统通过多个指标评估模型性能——如结构连贯性、真实性、召回精度、相关性、内容深度、覆盖广度、新颖性、冗余性、幻觉频率和事实准确性——利用大规模语言模型和基于规则的评估。详细描述可见表格 5 。

上下文管理器：多步强化学习的内在复杂性需要强大而动态的上下文管理。我们的上下文管理器包含三个基本功能：

- 提示更新：根据历史交互动态调整提示。
- 历史记录：保持互动的详细记录——包括提示、响应、处罚、检索操作和评估得分——以便精确重建学习轨迹。
- 优势分配：通过结合步骤级格式惩罚和轨迹级奖励来分配精细调整的优势分数，从而促进精确的令牌级优化。

并行环境交互：为了减少由异步外部依赖（如检索和评估系统）造成的延迟，我们使用专用服务器实例进行并行环境交互。受最近异步 API 驱动的多轮强化学习系统发展启发，这一策略通过减少反馈回路瓶颈显著提升了训练效率。

3.1.3 评估

基线 为验证我们模型的有效性，我们也检查了其相对于以下代表性基线系统的性能。这些系统被选择以涵盖已建立方法的广泛范围，并提供全面的比较基准：(1) Naive RAG 是一种简单的检索增强生成方法。它直接将所有查询检索到的文档输入到模型中，模型随后在一次非迭代式的过程中生成一份全面的综述。(2) AutoSurvey (Wang et al., 2024c) 是一个结构化的自动学术综述生成框架，通常采用系统化的过程，如规划、多源文献检索以及一致内容合成来产生输出。(3) WebThinker (Li et al., 2025b) 是一个由大型推理模型驱动的深度研究框架，用于像问答或报告生成这样的任务。我们使用 QwQ-32B (Team, 2025) 和 DeepSeek-R1-Distill-Qwen-7B 模型作为比较基线来评价免训练配置。(4) OpenAI Deep Research 是一个使用多步骤在线信息获取来从用户查询生成详细报告的 OpenAI

Table 6: 调查生成系统的性能对比。“G2FT”代表 Gemini-2.0-Flash-Thinking，“WTR1-7B”表示 Webthinker-R1-7B。由于 Webthinker 不包含引用功能，FactScore 评估被省略；对于 OpenAI Deep Research，因其在导出结果时不提供引用，也未进行 FactScore 评估。

Method	Content Quality					Faithfulness
	Relevance	Coverage	Depth	Novelty	Avg.	Fact Score
Naive RAG (driven by G2FT)	3.25	2.95	3.35	2.60	3.04	43.68
AutoSurvey (driven by G2FT)	3.10	3.25	3.15	3.15	3.16	46.56
Webthinker (driven by WTR1-7B)	3.30	3.00	2.75	2.50	2.89	–
Webthinker (driven by QwQ-32B)	3.40	3.30	3.30	2.50	3.13	–
OpenAI Deep Research (driven by GPT-4o)	3.50	3.95	3.55	3.00	3.50	–
MiniCPM4-Survey	3.45	3.70	3.85	3.00	3.50	68.73
w/o RL	3.55	3.35	3.30	2.25	3.11	50.24

系统，利用 GPT 的长形式推理能力。注意 AutoSurvey 和 Naive RAG 都是无训练方法。我们使用 `gemini-2.0-flash-thinking- exp-1219` [9] 作为它们的骨架。

评估细节　我们使用由 Wang et al. (2025a) 发布的 SurveyEval 数据集作为测试集，其中包括 20 个测试示例。受 STORM (Shao et al., 2024a) 和 FactScore (Min et al., 2023) 的启发，我们使用以下四个指标来评估模型生成调查的质量：(1) *相关性* 评估调查是否有效地保持了对用户特定查询的清晰关注，并与其直接相关，避免了不相关的偏题或离题。(2) *覆盖* 评估调查是否全面覆盖指定主题，彻底探索其关键子领域、多样化方面，以及领域内的重要已有知识。(3) *深度* 确定调查是否彻底审视核心主题及其相关领域，提供足够的分析细节、关键评价以及对复杂隐含问题的有意义见解。(4) *新颖性* 判断调查是否引入真正的新颖观点、原创诠释或与用户初始查询相关的重要但之前未表达的联系。(5) *事实评分* 量化调查中所呈现的离散、可验证原子事实的比例，这些事实是否被适当且可信的引用文献准确明确地证实。我们使用 GPT-4o 作为评估这些指标的判断者。

结果　如表 6 所示，我们提出的方法在内容相关的指标上优于由开源（例如，Webthinker）和闭源模型（例如，AutoSurvey）驱动的基线系统。我们的方法实现了与 OpenAI 深度研究相当的性能，突显其有效性和竞争力。此外，在已考查的系统中，我们的方法在事实指标上取得了最高分。此外，MiniCPM4-Survey 从 SFT 到 RL 显示出显著的改进，强调了在后期阶段引入的 RL 组件的功效。具体来说，在覆盖率、深度和新颖性上都有所提升，表明探索和规划能力的增强。尽管取得了这些进步，MiniCPM4-Survey 在覆盖率和新颖性方面仍落后于一些基线方法，表明在强化其战略规划和探索能力方面还有进一步提升的空间。

3.2 MiniCPM4 -MCP: 使用模型上下文协议的工具

大型语言模型（LLMs）与外部工具之间的交互逻辑传统上是静态设计且缺乏标准化，这与代理和工具的快速、独立演进不兼容。这种不兼容导致了高昂的维护成本、较差的可扩展性、有限的先前交互模式的可重用性以及碎片化的设计标准，最终导致了冗余的开发工作 (Hou et al., 2025)。为了解决这个问题，MCP 建立了一个通用和标准化的框架，使 LLMs 能够与多种工具无缝且安全地连接，从而促进各种资源的协调利用。

最近，开放源社区中已经构建了各种带有工具的 MCP 服务器 (Hou et al., 2025)，旨在使 LLM 能够发现、选择和编排不同的工具，以满足现实世界任务解决的需求。为了配合 MCP 服务器的快速发展，我们研究了使 MiniCPM4 具备 MCP 工具调用能力的潜力。

为此，我们提出了 MiniCPM4 -MCP，这是一个建立在 MiniCPM4 -8B 基础上的模型，能够通过 MCP 与各种工具和数据资源交互来解决广泛的实际任务。我们展示了如何调整 MiniCPM4 以掌握配备工具的 MCP 服务器。在我们人工标注的 MCP 工具调用测试数据上的总体模型性能展示了 MiniCPM4 -MCP 的有效性。

[9] https://ai.google.dev/gemini-api/docs/models

3.2.1 数据构建

我们的数据构建过程包括三个主要部分，包括数据生成、反向数据生成以及将现有函数调用数据集转换为 MCP 工具所使用的格式，其中反向数据生成包含单工具和跨工具设置。所有数据均经过人工和 LLM 辅助的质量检查程序。更多详情如下。

数据生成　许多现有数据集包含高质量的查询以及最终结果的注释。我们专注于识别那些可以通过 MCP 服务器有效解决的查询。为此，我们手动检查了公开可用数据集所涵盖的场景，并选择了相关的子集。对于每个选定的数据集，我们使用集成了 LLM 和 MCP 服务器的环境中的客户端进行查询完成过程的采样。保留与原始注释一致的最终结果的轨迹，并用于构建训练数据。

反向数据生成　对于集成在环境中的服务器和工具，我们使用 Claude-3.7-Sonnet 基于每个工具的描述执行反向查询构建。具体来说，Claude-3.7-Sonnet 生成需要使用目标工具的查询，然后调用该工具来解决构建的查询。此过程产生单一工具数据实例，被称为单一工具数据的反向构建。此外，为了训练 MiniCPM4 的跨工具调用能力，我们通过选择同一服务器下的两个工具及其各自的描述来构建跨工具数据。Claude-3.7-Sonnet 然后生成需要同时使用这两个工具的查询，并被限制调用两个指定的工具来解决这些查询。

从现有工具学习数据的转换　我们收集了与工具使用相关的公开可用数据集，并提取了可以解析并转换为符合 MCP 工具调用模式的轨迹格式的数据。这些数据用于训练 MiniCPM4 ，旨在使其具备基本功能，例如符合工具调用格式，准确解释用户指令，正确填充指定参数。该数据集大约包含 14 万个实例。

所有构建的数据都会经历双重阶段的质量检查，包括人工评估和大型语言模型（LLM）的验证。通过人工检查的轨迹优先作为测试数据，而那些通过 LLM 检查但未经人工验证的轨迹则保留为训练数据。

3.2.2 训练策略

我们主要采用从示范中学习的方法来训练我们的模型。这些示范是通过大型语言模型（LLM）与 MCP 环境之间的持续交互生成的。因此，在本节中，我们首先介绍环境构建中的关键特性。然后，我们展示与环境交互的客户端的关键特性。最后，我们描述 MiniCPM 如何从这些示范中学习。

MCP 环境构建　MCP 环境的设置涉及大量工作和漫长的调试过程。为了减少开发人员的工作量并实现即插即用配置，我们采用了基于 Docker 的方法，其中所有服务器都安装在 Docker 容器中。具体而言，我们从官方和社区 MCP 网站收集常用和流行的 MCP 服务器，涵盖办公生产力、日常生活、通信、信息服务和工作管理等各个领域。然后，这些服务器在 Docker 容器中进行调试，直到它们可以成功启动。

MCP 环境交互　在 MCP 中，客户端组件负责与 MCP 服务器进行交互，包括与服务器进行握手以获取可用工具列表。然而，市场上的许多 MCP 服务器是由社区开发者独立部署的，并且没有经过严格测试或质量保证。直接将所有列出的工具暴露给 LLM 可能导致频繁的工具调用失败，从而干扰任务的执行并损害模型的性能。为了解决这个问题，我们在客户端中加入了一个工具质量检查机制。具体来说，我们提示 Claude-4 生成 10 个查询，每个查询需要根据给定工具的描述和元数据调用当前工具。如果通过 LLM 执行这些查询返回工具的响应，无论响应是否正确，这表明与工具的通信正常运行。在此测试中实现 100 % 成功率的工具与其对应的服务器一起暴露给 LLM。

基于 MCP 客户端的　演示学习，LLMs（例如，MiniCPM4 -MCP）与 Docker 容器通信，实现 LLMs 与构建环境之间的无缝互动。我们采用配备强大 LLM 的客户端与我们构建的环境进行互动。累积的互动经验经过筛选，形成 MiniCPM4 的 SFT 训练数据。

3.2.3 评估

评估细节　根据现有通用评估指标 (Qin et al., 2024) 的做法，我们评估了在我们人工标注的测试数据中每个工具调用的工具名称、参数名称和参数值的准确性。对于那些包含多个工具调用步骤的跟踪，我们通过提供之前的真实步骤来评估当前步骤的准确性。

结果　我们的人为标注 MCP 工具调用测试数据上模型表现的总体结果如表 7 所示。根据实验结果，我们发现 Qwen3-8B 具备基本的 MCP 工具调用能力，展示出了对 MCP 工具的理解和使用能力，主要是因为调用 MCP 工具的经验与调用常规工具的经验有很大的重叠。然而，当涉及到较新的工具或更专业领域的工具（例如，arXiv，airbnb 等）时，Qwen3 显得较不熟悉。它倾向于在生成参数名

Table 7: MCP 工具使用准确率 (%)，其中 “func”、“param” 和 “p_v” 分别表示工具调用的函数名称、参数名称和参数值。“Average” 代表跨 MCP 服务器的样本加权平均准确率。

	GPT-4o			Qwen3 8B			MiniCPM4-MCP		
MCP Servers	func	param	p_v	func	param	p_v	func	param	p_v
Airbnb	89.3	67.9	53.6	92.8	60.7	50.0	96.4	67.9	50.0
Amap-Maps	79.8	77.5	50.0	74.4	72.0	41.0	89.3	85.7	39.9
Arxiv-MCP-Server	85.7	85.7	85.7	81.8	54.5	50.0	57.1	57.1	52.4
Calculator	100.0	100.0	20.0	80.0	80.0	13.3	100.0	100.0	6.67
Computor-Control-MCP	90.0	90.0	90.0	90.0	90.0	90.0	90.0	90.0	86.7
Desktop-Commander	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Filesystem	63.5	63.5	31.3	69.7	69.7	26.0	83.3	83.3	42.7
Github	92.0	80.0	58.0	80.5	50.0	27.7	62.8	25.7	17.1
Gaode	71.1	55.6	17.8	68.8	46.6	24.4	68.9	46.7	15.6
MCP-Code-Executor	85.0	80.0	70.0	80.0	80.0	70.0	90.0	90.0	65.0
MCP-Docx	95.8	86.7	67.1	94.9	81.6	60.1	95.1	86.6	76.1
PPT	72.6	49.8	40.9	85.9	50.7	37.5	91.2	72.1	56.7
PPTx	64.2	53.7	13.4	91.0	68.6	20.9	91.0	58.2	26.9
Simple-Time-Server	90.0	70.0	70.0	90.0	90.0	90.0	90.0	60.0	60.0
Slack	100.0	90.0	70.0	100.0	100.0	65.0	100.0	100.0	100.0
Whisper	90.0	90.0	90.0	90.0	90.0	90.0	90.0	90.0	30.0
Average	80.2	70.2	49.1	83.5	67.7	43.8	88.3	76.1	51.2

称和传递参数值时套用其他工具的知识，而未能充分调整或适应特定 MCP 工具的要求。相比之下，MiniCPM4 从示范中学习，从而了解我们收集的 MCP 服务器和工具的特性，这使得其在测试数据上的表现更好。

在这份技术报告中，我们展示了 MiniCPM4 ，其特点是高效的预训练和推理。得益于高效的预训练数据和基础设施，我们仅使用 8 万亿个标记便能达到与现有开源模型相当的性能。而通过高效的架构和推理系统，我们可以在长序列处理上实现 5× 的加速。为了促进开源社区的发展，我们发布了 MiniCPM4 的模型参数和推理代码。

在未来，我们将继续研究 LLMs 的高效训练和推理。在模型架构方面，我们将致力于高效的稀疏模型架构，目标是使 LLMs 能够在终端设备上处理无限长的序列。在数据构建方面，我们的研究将重点提高现有语料库的质量，并合成大规模的推理密集型预训练数据集，这可以显著提升 LLMs 的基础能力。此外，我们将继续探索强化学习的巨大潜力，以便 LLMs 能从各种环境中学习技能。至于推理系统，我们计划为大多数终端平台开发高效的系统，以帮助社区运行和评估我们的模型。

4 贡献与致谢

MiniCPM4 是我们团队所有成员共同努力的结果。

项目设计和协调　肖超俊、李宇轩、韩旭

贡献者（按姓氏排序）　白榆卓，蔡杰，陈浩天，陈文桐，丛鑫，崔甘屈，丁宁，范圣丹，方烨巍，符子萱，关文与，关怡同，郭君绍，涵宇峰，何冰祥，黄宇翔，孔存亮，李秋佐，李思远，李文浩，李阳昊，李宜善，李振，刘丹，林碧苑，林彦凯，龙翔，路群瑛，吕雅曦，罗培岩，吕虹雅，瓯力图，潘殷旭，屈则凯，史群栋，宋子俊，苏嘉苑，苏周，孙傲，孙祥辉，唐沛军，王方正，王峰，王硕，王煜东，吴叶赛，肖振宇，谢杰，谢子豪，燕玉坤，袁佳锐，张凯霍，张磊，张林悦，张雪仁，张余帝，赵恒宇，赵维林，赵伟伦，赵远谦，郑智，周戈，周杰，周伟，周紫涵，周子萱

监督　韩旭、刘志远、曾国阳、贾超、李大海、孙茂松

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Proceedings of NeurIPS*, 37:100213–100240, 2024.

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

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. LongAlign: A recipe for long context alignment of large language models. In *Findings of ACL: EMNLP 2024*, pp. 1376–1395, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.74.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal lr across token horizons. *arXiv preprint arXiv:2409.19913*, 2024.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *Proceedings of ICLR*, 2023.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of NeurIPS*, 2020.

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

Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. Prefixquant: Static quantization beats dynamic through prefixed outliers in llms. *arXiv preprint arXiv:2410.05265*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Proceedings of NeurIPS*, 35:16344–16359, 2022.

Team DeepSeek, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Team DeepSeek, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of EMNLP*, pp. 3029–3051, 2023.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending LLM context window beyond 2 million tokens. *CoRR*, abs/2402.13753, 2024. doi: 10.48550/ARXIV.2402.13753.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Katie E Everett, Lechao Xiao, Mitchell Wortsman, Alexander A Alemi, Roman Novak, Peter J Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, et al. Scaling exponents across parameterizations and optimizers. In *Proceedings of ICML*, pp. 12666–12700. PMLR, 2024.

Clémentine Fourrier, Nathan Habib, Hynek Kydlíek, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL https://github.com/huggingface/lighteval.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. In *Proceedings of ICLR*, 2023.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of ICML*, pp. 15706–15734, 2024.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. Apple intelligence foundation language models. *CoRR*, abs/2407.21075, 2024. doi: 10.48550/ARXIV.2407.21075.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2020.

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. In *Proceedings of NeurIPS: Datasets and Benchmarks Track*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions, April 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. Ruler: What' s the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *CoRR*, abs/2404.06395, 2024. doi: 10.48550/ARXIV.2404.06395.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Proceedings of NeurIPS*, 36:62991–63010, 2023.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference: Accelerating pre-filling for long-context llms via dynamic sparse attention. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *Proceedings of ICLR*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

Tanishq Kumar, Zachary Ankner, Benjamin F. Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision, 2024.

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. In *Findings of ACL: ACL 2024*, pp. 11260–11285, 2024a.

Houyi Li, Wenzheng Zheng, Jingcheng Hu, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i – optimal hyperparameter scaling law in large language model pretraining, 2025a.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024b.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*, 2025b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of EMNLP*, pp. 7421–7432, 2024c.

Feng Lin, Hanling Yi, Hongbin Li, Yifan Yang, Xiaotian Yu, Guangming Lu, and Rong Xiao. Bita: Bi-directional tuning for lossless acceleration in large language models, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of MLSys*, 6:87–100, 2024b.

Lucas Lingle. A large-scale exploration of μ-transfer. *arXiv preprint arXiv:2404.05728*, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of ACL: ACL 2024*, pp. 467–484, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.26.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Proceedings of NeurIPS*, 37:54463–54482, 2024c.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Shuming Ma, Hongyu Wang, Shaohan Huang, Xingxing Zhang, Ying Hu, Ting Song, Yan Xia, and Furu Wei. Bitnet b1.58 2b4t technical report, 2025.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*, 2023.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. *Technical Report*, 2024a. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI. Openai o1 system card. *CoRR*, abs/2412.16720, 2024b.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025.

Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of NeurIPS*, 2022.

Xu Ouyang, Tao Ge, Thomas Hartvigsen, Zhisong Zhang, Haitao Mi, and Dong Yu. Low-bit quantization favors undertrained llms: Scaling laws for quantized llms with 100t training tokens, 2024.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Proceedings of NeurIPS*, 37:30811–30849, 2024.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *CoRR*, abs/2309.00071, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *Proceedings of ICLR*, 2024.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271, 2020.

Yijia Shao, Yucheng Jiang, Theodore A Kanell, Peter Xu, Omar Khattab, and Monica S Lam. Assisting in writing wikipedia-like articles from scratch with large language models. *arXiv preprint arXiv:2402.14207*, 2024a.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024b.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. In *Proceedings of CoLM*, 2024a.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024b.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of ACL: ACL 2023*, 2023.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Thomas van Dongen and Stephan Tulkens. Semhash: Fast semantic text deduplication & filtering, 2025. URL https://github.com/MinishLab/semhash.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NeurIPS*, pp. 5998–6008, 2017.

Haoyu Wang, Yujia Fu, Zhu Zhang, Shuo Wang, Zirui Ren, Xiaorong Wang, Zhili Li, Chaoqun He, Bo An, Zhiyuan Liu, and Maosong Sun. Llm×mapreduce-v2: Entropy-driven convolutional test-time scaling for generating long-form articles from extremely long resources, 2025a.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

Liangdong Wang, Bo-Wen Zhang, Chengwei Wu, Hanyu Zhao, Xiaofeng Shi, Shuhao Gu, Jijie Li, Quanyue Ma, TengFei Pan, and Guang Liu. Cci3. 0-hq: a large-scale chinese dataset of high quality designed for pre-training large language models. *arXiv preprint arXiv:2410.18505*, 2024a.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Proceedings of ICML*, 2024b.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Qingsong Wen, Wei Ye, et al. Autosurvey: Large language models can automatically write surveys. *Advances in Neural Information Processing Systems*, 37:115119–115145, 2024c.

Yudong Wang, Zixuan Fu, Jie Cai, Peijun Tang, Hongya Lyu, Yewei Fang, Zhi Zheng, Jie Zhou, Guoyang Zeng, Chaojun Xiao, et al. Ultra-fineweb: Efficient data filtering and verification for high-quality llm training data. *arXiv preprint arXiv:2505.05427*, 2025b.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *TMLR*, 2022.

Bingquan Xia, Bowen Shen, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang, Huaqiu Liu, Jiebao Xiao, Jinhao Dong, Liang Zhao, et al. Mimo: Unlocking the reasoning potential of language model–from pretraining to posttraining. *arXiv preprint arXiv:2505.07608*, 2025.

Chaojun Xiao, Jie Cai, Weilin Zhao, Guoyang Zeng, Biyuan Lin, Jie Zhou, Zhi Zheng, Xu Han, Zhiyuan Liu, and Maosong Sun. Densing law of llms. *arXiv preprint arXiv:2412.04315*, 2024a.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In *Proceedings of NeurIPS*, 2024b.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *CoRR*, abs/2309.17453, 2023.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. Onebit: Towards extremely low-bit large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of AAAI*, volume 34, pp. 6672–6679, 2020.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025a.

Yijiong Yu, Ziyun Dai, Zekun Wang, Wei Wang, Ran Chen, and Ji Pei. Opencsg chinese corpus: A series of high-quality chinese datasets for llm training. *arXiv preprint arXiv:2501.08197*, 2025b.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025a.

Yudi Zhang, Weilin Zhao, Xu Han, Tiejun Zhao, Wang Xu, Hailong Cao, and Conghui Zhu. Speculative decoding meets quantization: Compatibility evaluation and hierarchical framework design. *arXiv preprint arXiv:2505.22179*, 2025b.

Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. Qspec: Speculative decoding with complementary quantization schemes. *arXiv preprint arXiv:2410.11305*, 2024.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Ao Sun, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jianyong Wang, et al. Fr-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. *arXiv preprint arXiv:2502.14856*, 2025.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*, 2024.