本文由 AINLP 公众号整理翻译，更多 LLM 资源请扫码关注!

Llama-Nemotron: Efficient Reasoning Models

NVIDIA

Abstract

We introduce the Llama-Nemotron series of models, an open family of heterogeneous reasoning models that deliver exceptional reasoning capabilities, inference efficiency, and an open license for enterprise use. The family comes in three sizes—Nano (8B), Super (49B), and Ultra (253B)—and performs competitively with state of the art reasoning models such as DeepSeek-R1 while offering superior inference throughput and memory efficiency. In this report, we discuss the training procedure for these models, which entails using neural architecture search from Llama 3 models for accelerated inference, knowledge distillation, and continued pretraining, followed by a reasoning-focused post-training stage consisting of two main parts: supervised fine-tuning and large scale reinforcement learning. Llama-Nemotron models are the first open-source models to support a dynamic reasoning toggle, allowing users to switch between standard chat and reasoning modes during inference. To further support open research and facilitate model development:

- We release the Llama-Nemotron reasoning models—LN-Nano, LN-Super, and LN-Ultra—under the commercially permissive NVIDIA Open Model License Agreement.
  - `Llama-3.1-Nemotron-Nano-8B-v1` 🤗
  - `Llama-3.3-Nemotron-Super-49B-v1` 🤗
  - `Llama-3.1-Nemotron-Ultra-253B-v1` 🤗
  - `Llama-3.1-Nemotron-Ultra-253B-CPT-v1` 🤗
- We release the **complete post-training dataset**.
  - `Llama-Nemotron-Post-Training-Dataset` 🤗
- We also release our training codebases: `NeMo`, `NeMo-Aligner`, `Megatron-LM`.

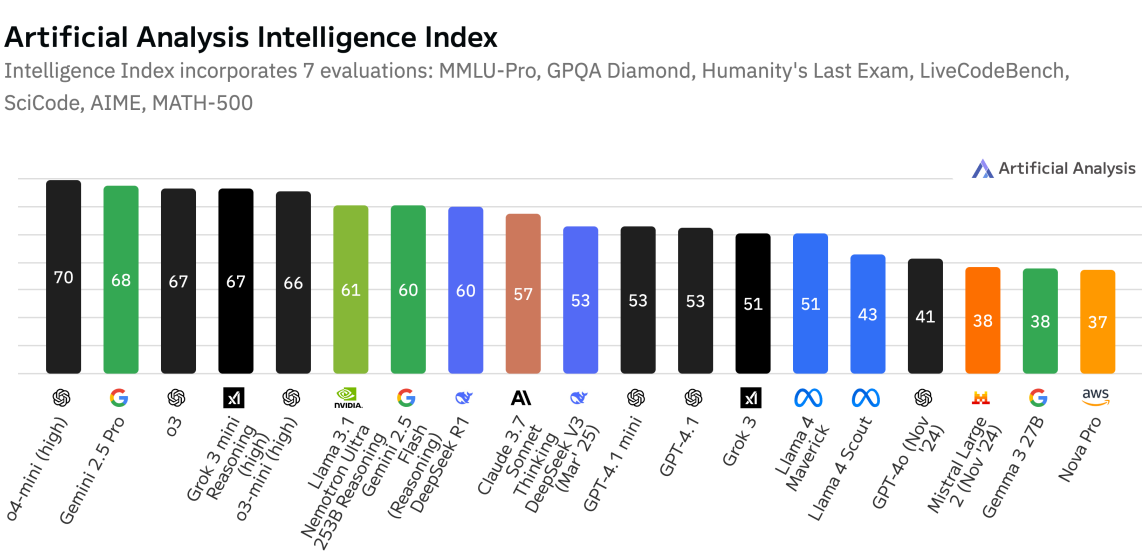


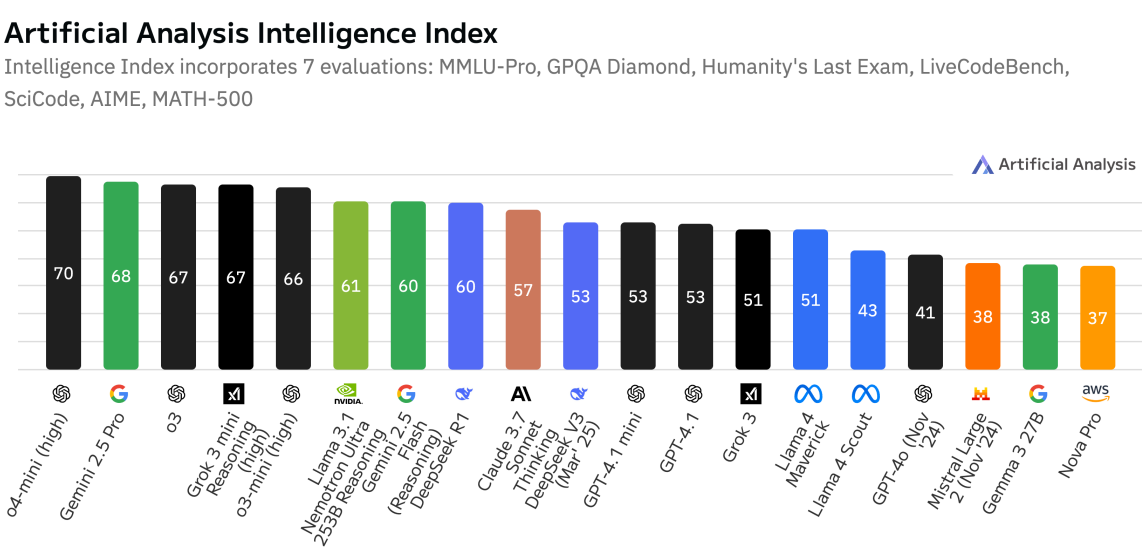
Figure 1 | As of April 2025, our flagship model LN-Ultra is the most "intelligent" open model according to Artificial Analysis.

Llama-Nemotron：高效推理模型

NVIDIA

摘要
我们介绍Llama-Nemotron系列模型，这是一系列开源的异构推理模型，具有卓越的推理能力、推理效率以及面向企业使用的开放许可证。该系列包括三种尺寸——Nano（8B）、Super（49B）和Ultra（253B）——在性能上与最先进的推理模型如DeepSeek-R1相竞争，同时提供更优的推理吞吐量和内存效率。在本报告中，我们讨论了这些模型的训练过程，包括利用Llama 3模型的神经架构搜索以加速推理、知识蒸馏和持续预训练，随后进行以推理为重点的后训练阶段，主要包括两个部分：监督微调和大规模强化学习。Llama-Nemotron模型是首批支持动态推理切换的开源模型，允许用户在推理过程中在标准聊天和推理模式之间切换。为了进一步支持开源研究并促进模型开发：

- 我们发布了Llama-Nemotron推理模型——LN-Nano、LN-Super和LN-Ultra——根据具有商业宽容性的NVIDIA开放模型许可协议。
  - `Llama-3.1-Nemotron-Nano-8B-v1` 🤗
  - `Llama-3.3-Nemotron-Super-49B-v1` 🤗
  - `Llama-3.1-Nemotron-Ultra-253B-v1` 🤗
  - `Llama-3.1-Nemotron-Ultra-253B-CPT-v1` 🤗
- 我们发布了完整的后训练数据集。
  - `Llama-Nemotron-Post-Training-Dataset` 🤗
- 我们还发布了我们的训练代码库：`NeMo`，`NeMo-Aligner`，`Megatron-LM`。



F图 1 | 截至 2025 年 4 月，我们的旗舰型号 LN-Ultra 是最“智能”的开源模型
a根据人工分析。

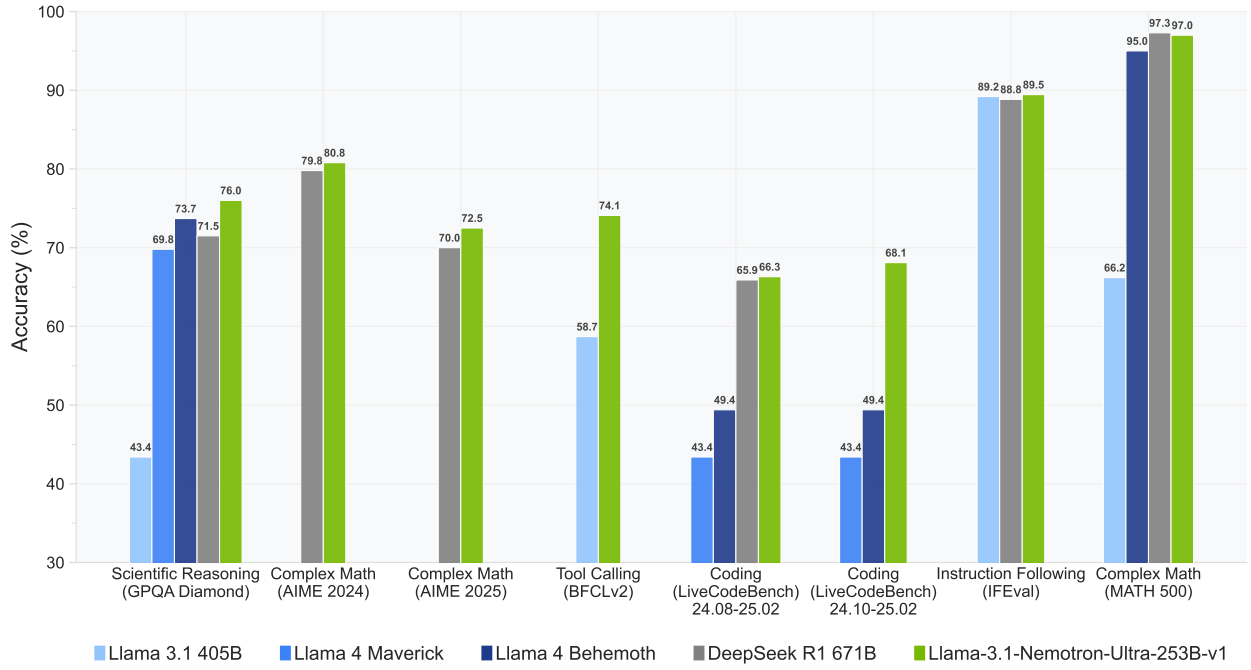


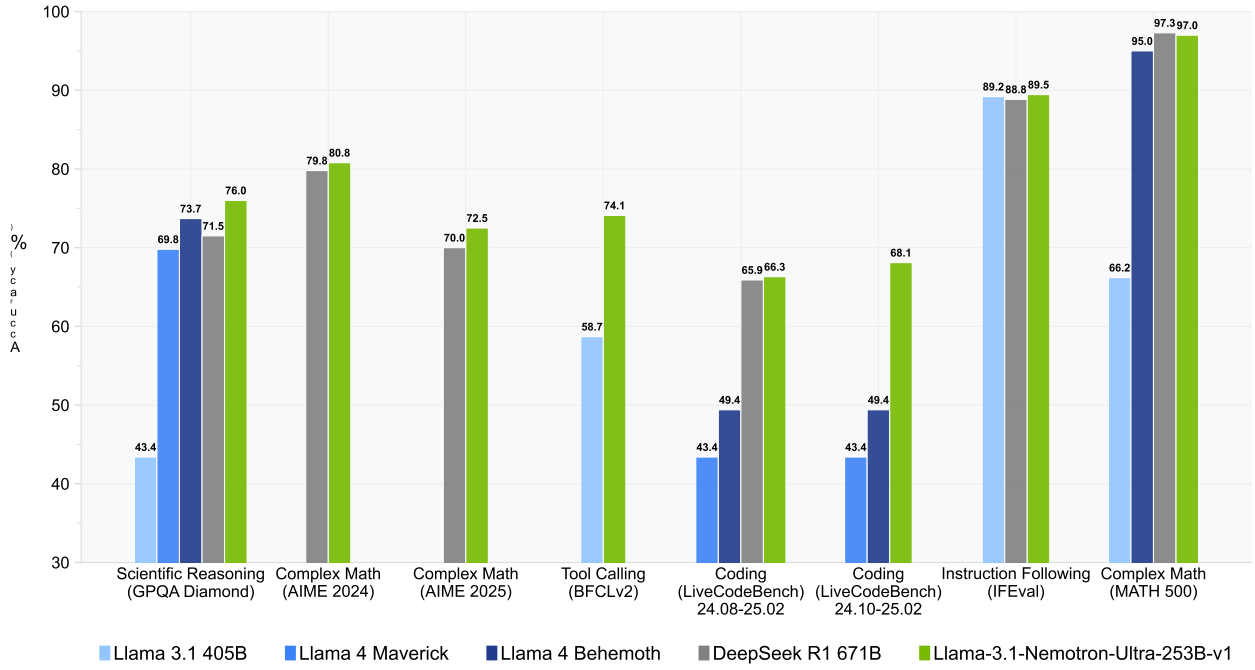
Figure 2 | LN-Ultra delivers leading performance among open models across a wide range of reasoning and non-reasoning benchmarks.

1. Introduction

In recent years the pace of language model development has been increasing, leading to rapid improvements in performance across a wide range of natural language processing tasks. Most recently, the introduction of reasoning models such as OpenAI o1 (OpenAI, 2025) and DeepSeek-R1 (DeepSeek-AI et al., 2025) has marked a new phase of advancement, resulting in models that can think deeply about problems before answering. A defining characteristic of these models is their long responses, often containing long chains of thought, self-verification, reflection, and backtracking. Such long responses enable them to achieve state-of-the-art performance across a wide variety of tasks, including PhD-level STEM questions and competition-level math problems.

As reasoning capabilities increasingly depend on scaling at inference time, it has become essential to design models that are efficient to run during inference. Inference efficiency is no longer just a deployment concern—it is now a core limiting factor for overall model intelligence and the viability of agentic pipelines. As such, maximizing inference efficiency is a primary optimization objective for these models. Beyond raw inference efficiency, it is equally critical to expose control over reasoning behavior to the end user. Not all queries benefit from detailed multi-step reasoning—such responses may be unnecessarily verbose or even counterproductive in certain contexts. Granting users the ability to toggle reasoning on or off ensures that inference resources are allocated judiciously and that response styles remain appropriate to the task (Anthropic, 2025).

In this paper, we detail the training of the Llama-Nemotron (LN) family of models, an open family of heterogeneous reasoning models that deliver exceptional reasoning capabilities, inference efficiency, and an open license for enterprise use. The models come in three sizes— LN-Nano (8B), LN-Super (49B), and LN-Ultra (253B). Notably, LN-Ultra outperforms DeepSeek-R1 while fitting on a single 8xH100 node and achieving higher inference throughput. These models are derived from Llama 3.1 and Llama 3.3 (Grattafiori et al., 2024), and are optimized for high-throughput inference while



F图2 | LN-Ultra 在各种推理任务中在开源模型中表现领先
and 非推理基准。

1. 引言

近年来，语言模型的发展速度不断加快，导致在各种自然语言处理任务中的性能迅速提升。最近，诸如OpenAI o1（OpenAI，2025）和DeepSeek-R1（DeepSeek-AI等，2025）等推理模型的引入标志着一个新的发展阶段，产生了能够在回答前深入思考问题的模型。这些模型的一个显著特征是它们的长回答，通常包含长长的思维链、自我验证、反思和回溯。这种长回答使它们能够在包括博士级别的STEM问题和竞赛级数学题在内的各种任务中实现最先进的性能。

随着推理能力越来越依赖于推理时的扩展，设计在推理过程中高效运行的模型变得至关重要。推理效率不再仅仅是部署方面的关注点——它现在已成为整体模型智能和代理管道可行性的核心限制因素。因此，最大化推理效率成为这些模型的主要优化目标。除了原始的推理效率之外，同样关键的是向最终用户暴露对推理行为的控制。并非所有查询都适合详细的多步骤推理——在某些情况下，这样的回答可能过于冗长甚至适得其反。赋予用户切换推理开启或关闭的能力，确保推理资源得到合理分配，并且响应风格与任务保持一致（Anthropic, 2025）。

在本文中，我们详细介绍了Llama-Nemotron（LN）系列模型的训练过程，这是一系列开源的异构推理模型，具有卓越的推理能力、推理效率，并提供企业使用的开放许可证。这些模型有三种尺寸——LN-Nano（8B）、LN-Super（49B）和LN-Ultra（253B）。值得注意的是，LN-Ultra在只需一个8x H100节点的情况下，性能优于DeepSeek-R1，并实现了更高的推理吞吐量。这些模型源自Llama 3.1和Llama 3.3（Grattafiori等，2024），并经过优化，以实现高吞吐量推理，同时

delivering strong reasoning performance and a context length of 128K tokens. Each model supports a reasoning toggle that lets users dynamically switch between standard chat and reasoning modes at inference time using a lightweight system prompt: `"detailed thinking on/off"`. This design enables both cost-effective general-purpose use and detailed multi-step reasoning, without requiring separate models or architectures.

The Llama-Nemotron models are constructed in five stages. The first stage consists of optimizing inference efficiency with neural architecture search (NAS) from the Llama 3 series of models and applying Feed-Forward Network (FFN) Fusion. The second stage includes recovery training with knowledge distillation and continued pretraining. The third stage is supervised fine-tuning (SFT) on a mix of standard instruction data and reasoning traces from strong teachers such as DeepSeek-R1, which enables the model to perform multi-step reasoning. The fourth stage involves large-scale reinforcement learning on complex mathematics and STEM datasets, a crucial step for enabling the student model to surpass its teacher's capabilities. For LN-Ultra, this phase yields a substantial performance boost on the GPQA-D benchmark, cementing it as the best open-source model for scientific reasoning. To enable such large-scale RL training, we develop a custom training framework that contains a number of optimizations, most notably generation in FP8. The final stage is a short alignment phase focused on instruction following and human preference.

As part of this release, we also open-source the Llama-Nemotron-Post-Training-Dataset, a carefully curated dataset used during the supervised and reinforcement learning stages of training for LN-Nano, LN-Super, and LN-Ultra. It is designed to target key capabilities such as mathematical reasoning, coding, science, and instruction following, and consists of synthetic responses generated by a range of open-source models. Prompts and responses are filtered for quality, correctness, and complexity to provide strong training signals across a diverse set of tasks.

According to Artificial Analysis (shown in Figure 1), an independent benchmarking and analysis company focused on evaluating artificial intelligence models and API providers, LN-Ultra is the most intelligent open-sourced model as of April 2025. This release represents one of the largest contributions to the open source community in support of developing reasoning models.

2. Creating Inference-Optimized Models

The LN-Super and LN-Ultra models are optimized for efficient inference using the *Puzzle* framework (Bercovich et al., 2024). Puzzle is a neural architecture search (NAS) framework that transforms large language models into hardware-efficient variants under real-world deployment constraints, as illustrated in Figure 3. Starting from a Llama 3 Instruct model (Llama 3.3-70B-Instruct for LN-Super and Llama 3.1-405B-Instruct for LN-Ultra), Puzzle applies *block-wise local distillation* to build a library of alternative transformer blocks. Each block is trained independently and in parallel to approximate the function of its parent block while improving computational properties such as latency, memory usage, or throughput. This process allows each alternative block to approximate the original behavior with a certain accuracy-efficiency tradeoff profile; that is, some blocks in the library are more efficient but may incur some quality degradation—introducing an explicit tradeoff between computational cost and model accuracy. The block variants include:

- **Attention removal**: Some blocks omit the attention mechanism entirely, reducing both compute and KV-cache memory consumption.
- **Variable FFN dimensions**: The feed-forward network's intermediate size is varied, enabling compression at different granularity levels (e.g., 87%, 75%, 50%, down to 10% of the original hidden size).

提供强大的推理性能和128K标记的上下文长度。每个模型都支持一个推理切换开关，允许用户在推理时动态切换标准聊天和推理模式，使用一个轻量级的系统提示：`"detailed thinking on/off"`。这种设计实现了既经济实用的通用用途，又支持详细的多步骤推理，无需使用不同的模型或架构。

Llama-Nemotron模型分为五个阶段构建。第一阶段包括使用神经架构搜索（NAS）对Llama 3系列模型进行推理效率优化，并应用前馈网络（FFN）融合。第二阶段包括通过知识蒸馏进行恢复训练和持续预训练。第三阶段是在标准指令数据和来自强教师（如DeepSeek-R1）的推理轨迹的混合上进行有监督微调（SFT），使模型能够执行多步推理。第四阶段涉及在复杂数学和STEM数据集上的大规模强化学习，这是使学生模型超越其教师能力的关键步骤。对于LN-Ultra，这一阶段在GPQA-D基准测试中带来了显著的性能提升，巩固了其作为科学推理领域最佳开源模型的地位。为了实现如此大规模的RL训练，我们开发了一个定制的训练框架，包含多项优化，最显著的是在FP8中进行生成。最后阶段是一个简短的对齐阶段，专注于指令遵循和人类偏好。

作为此次发布的一部分，我们还开源了 Llama-Nemotron-Post-Training-Dataset，这是一个经过精心策划的数据集，用于 LN-Nano、LN-Super 和 LN-Ultra 训练中的监督学习和强化学习阶段。它旨在针对数学推理、编码、科学和指令执行等关键能力，由一系列开源模型生成的合成响应组成。提示和响应经过筛选，以确保质量、正确性和复杂性，为各种任务提供强有力的训练信号。

根据人工分析（如图1所示），一家专注于评估人工智能模型和API提供商的独立基准测试和分析公司，截止到2025年4月，LN-Ultra是最智能的开源模型。这一版本是对开源社区在支持开发推理模型方面的最大贡献之一。

2. 创建推理优化模型

LN-Super 和 LN-Ultra 模型针对使用 *Puzzle* 框架（Bercovich 等，2024）进行高效推理进行了优化。Puzzle 是一个神经架构搜索（NAS）框架，能够在实际部署约束下将大型语言模型转化为硬件高效的变体，如图 3 所示。从 Llama 3 指令模型（LN-Super 使用 Llama 3.3-70B-Instruct，LN-Ultra 使用 Llama 3.1-405B-Instruct）出发，Puzzle 应用 *block-wise local distillation* 构建一个替代变换器块的库。每个块都独立且并行训练，以逼近其父块的功能，同时改善计算性能，如延迟、内存使用或吞吐量。这个过程允许每个替代块以一定的准确性-效率权衡曲线逼近原始行为；也就是说，库中的一些块更高效，但可能会带来一定的质量下降——在计算成本和模型准确性之间引入明确的权衡。块变体包括：

- 注意力移除：一些块完全省略了注意力机制，减少了计算量和KV缓存内存的消耗。
- 变量FFN维度：前馈网络的中间层大小被调整，以实现不同粒度级别的压缩（例如，87%、75%、50%、直到原始隐藏层大小的10%）。

While Puzzle supports additional operations—including grouped-query attention (GQA) (Ainslie et al., 2023) with different numbers of key-value heads, linear alternatives to attention, and no-op substitutions—empirical evaluation showed that attention removal and FFN compression were the most effective for optimizing the LN-Super and LN-Ultra models in terms of overall throughput and memory savings.

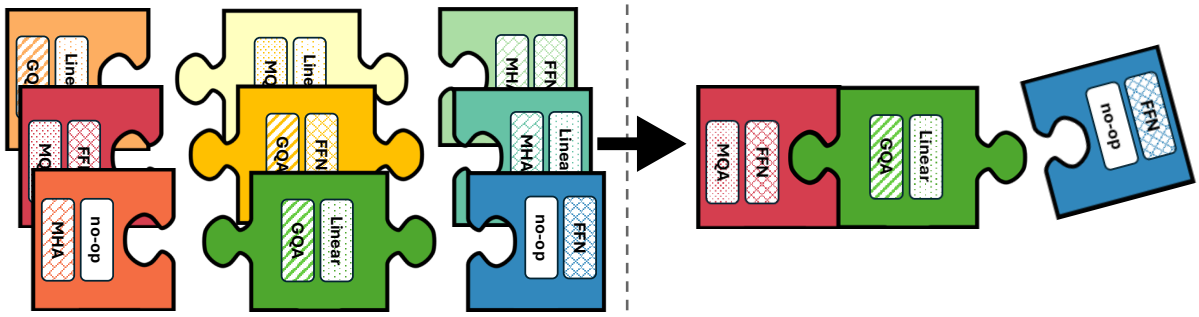


Figure 3 | Overview of the Puzzle framework.

Once the block library is built, Puzzle assembles a complete model by selecting one block per layer. This selection is governed by a mixed-integer programming (MIP) solver that identifies the most efficient configuration under a given set of constraints, such as hardware compatibility, maximum allowed latency, total memory budget, or desired inference throughput. Because Puzzle supports multiple block variants per layer with different accuracy-efficiency tradeoff profiles, it enables users to precisely target any point on the accuracy-efficiency Pareto frontier. For example, Puzzle can generate models that meet specific constraints relevant to agentic systems or deployment pipelines – such as bounded memory use or tight end-to-end response time.

Vertical Compression with FFN Fusion. For the LN-Ultra model, we introduce an additional compression technique called *FFN Fusion* (Bercovich et al., 2025), designed to reduce sequential depth and improve inference latency. This technique leverages a structural property that emerges after Puzzle removes some attention layers: the model often contains consecutive FFN blocks. FFN Fusion identifies such sequences and replaces them with fewer, wider FFN layers that can be executed in parallel. This reduces the number of sequential steps without compromising expressivity, and significantly improves compute utilization—especially on multi-GPU setups where inter-layer communication overhead is non-negligible.

2.1. Deployment Constraints and Efficiency Targets

LN-Super is optimized to run efficiently on a single NVIDIA H100 GPU with tensor parallelism 1 (TP1). Using Puzzle, we produce a model that achieves a 5× throughput speedup over Llama 3.3-70B-Instruct at batch size 256 and TP1. With one H100 GPU, even when Llama 3.3-70B-Instruct is run at its optimal configuration with TP4, LN-Super at TP1 still delivers a $\geq$2.17× throughput advantage. The model is also optimized under a constraint of approximately 300K cached tokens (batch size × sequence length), measured at FP8 precision on a single H100 GPU. For instance, this corresponds to processing batch size 16 and sequence length 18,750.

虽然Puzzle支持额外的操作——包括分组查询注意力（GQA）（Ainslie等，2023）与不同数量的键值头、注意力的线性替代方案以及无操作替换——但实证评估显示，去除注意力和FFN压缩在整体吞吐量和内存节省方面对优化LN-Super和LN-Ultra模型最为有效。

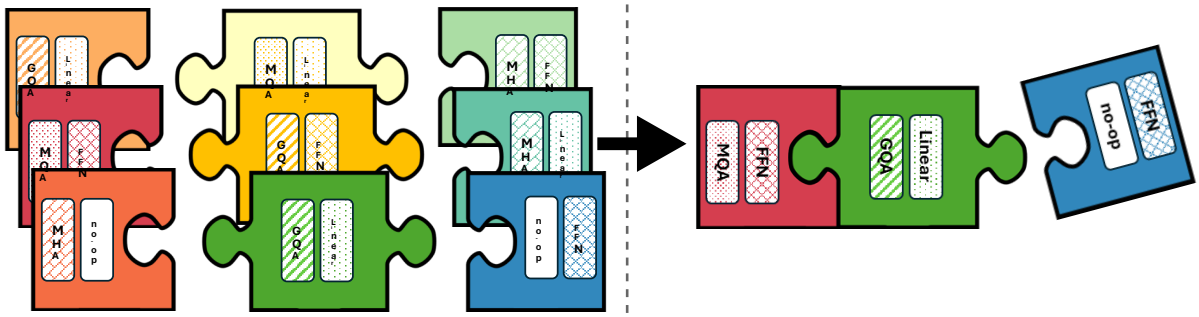


图 3 | 拼图框架概述。

一旦块库构建完成，Puzzle 就会通过为每一层选择一个块来组装完整的模型。这一选择由混合整数规划（MIP）求解器控制，它在一组给定的约束条件下（如硬件兼容性、最大允许延迟、总内存预算或期望的推理吞吐量）识别出最有效的配置。由于 Puzzle 支持每层多种具有不同准确率-效率权衡特征的块变体，它能够让用户精确地定位在准确率-效率帕累托前沿的任何点。例如，Puzzle 可以生成满足特定约束的模型，这些约束可能与智能系统或部署流程相关——比如有限的内存使用或紧凑的端到端响应时间。

使用FFN融合的垂直压缩。对于LN-Ultra模型，我们引入了一种额外的压缩技术，称为 *FFN Fusion* (Bercovich 等人，2025)，旨在减少序列深度并改善推理延迟。这项技术利用了Puzzle在移除一些注意力层后出现的结构特性：模型中常常包含连续的FFN块。FFN融合识别出这样的序列，并用更少、更宽的FFN层进行替换，这些层可以并行执行。这减少了序列步骤的数量，同时不影响表达能力，并显著提高了计算利用率——尤其是在多GPU设置中，层间通信开销不可忽视。

2.1. 部署限制和效率目标

LN-Super 被优化为在单个 NVIDIA H100 GPU 上以 tensor parallelism 1 (TP1) 高效运行。使用 Puzzle，我们生成的模型在批量大小为 256 和 TP1 时，比 Llama 3.3-70B-Instruct 提升了 5× 倍的吞吐速度。在一台 H100 GPU 上，即使在其最佳配置（TP4）下运行 Llama 3.3-70B-Instruct，LN-Super 在 TP1 下仍然具有 ≥2.17× 的吞吐优势。该模型还在大约 30 万缓存令牌（批量大小 ×，序列长度）限制下进行了优化，测量是在单个 H100 GPU 上以 FP8 精度进行。例如，这对应于处理批量大小 16 和序列长度 18,750。

LN-Ultra is optimized for a full H100 node (8 GPUs). During Puzzle's architecture search phase, the model is constrained to achieve at least a 1.5× latency reduction over Llama 3.1-405B-Instruct. After applying FFN Fusion, the final model achieves a 1.71× latency improvement. LN-Ultra is also optimized under cached tokens constraints, supporting up to 3M tokens at FP8 precision and 600K tokens at BF16 precision on an H100 node.

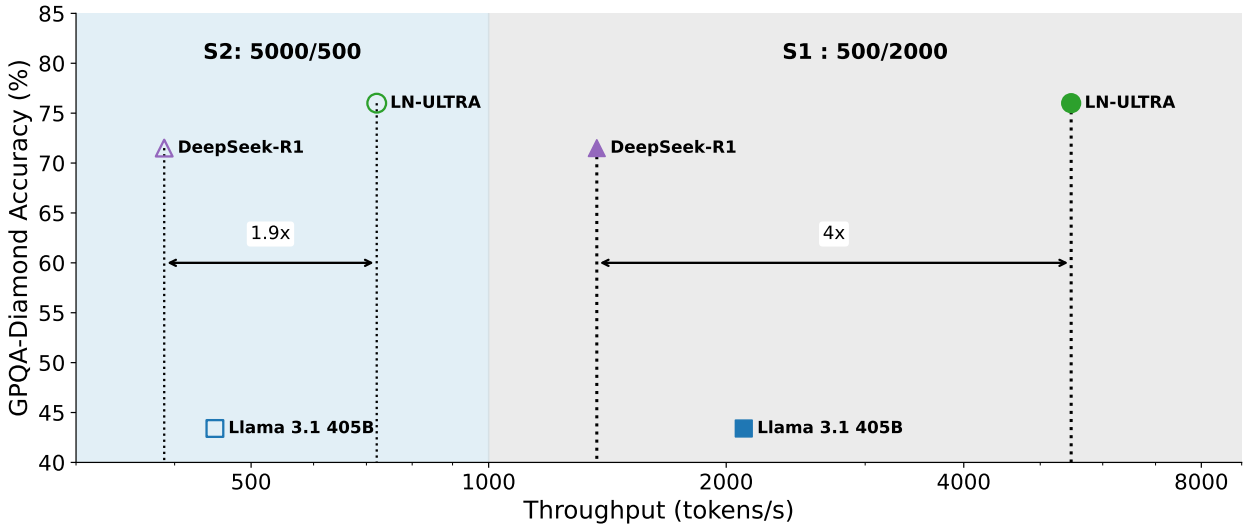


Figure 4 | GPQA-Diamond Accuracy vs. Throughput. We measure on two settings, S1: 500/2000 (ISL/OSL); S2: 5000/500 (ISL/OSL). Both with 250 concurrent users. Models are served with FP8. Note that we use 8×H100 for LN-Ultra and Llama 3.1 405B, but 8×H200 for Deepseek-R1 because of its size.

Figure 4 illustrates the trade-off between GPQA-Diamond accuracy (%) and processing throughput (tokens/s) for under two settings. Notably, LN-Ultra consistently outperforms the DeepSeek-R1 and Llama-3.1-405B in both accuracy and efficiency across these settings, clearly positioning it as a superior choice on the accuracy-throughput Pareto curve.

2.2. Post-NAS Training: Knowledge Distillation and Continued Pretraining

Following the NAS phase, both LN-Super and LN-Ultra undergo additional training to improve inter-block compatibility and recover any quality loss introduced during blockwise substitution.

- LN-Super is trained for 40B tokens using a knowledge distillation objective over the *Distillation Mix* dataset introduced by Bercovich et al. (2024).
- LN-Ultra is first trained with knowledge distillation for 65B tokens using the same distillation dataset, followed by 88B tokens of continued training on the Nemotron-H phase 4 pretraining dataset (NVIDIA et al., 2025).

This final pretraining step allows LN-Ultra to not only match but surpass the reference model Llama 3.1-405B-Instruct in key benchmarks, demonstrating that aggressive architecture optimization can be reconciled with high model performance through short distillation and pretraining (see Table 1).

LN-Ultra 针对完整的 H100 节点（8 个 GPU）进行了优化。在 Puzzle 的架构搜索阶段，模型被限制至少实现比 Llama 3.1-405B-Instruct 低 1.5× 的延迟。在应用 FFN 融合后，最终模型实现了 1.71× 的延迟提升。LN-Ultra 也在缓存令牌限制下进行了优化，支持在 H100 节点上以 FP8 精度处理最多 3M 令牌，以及以 BF16 精度处理 600K 令牌。

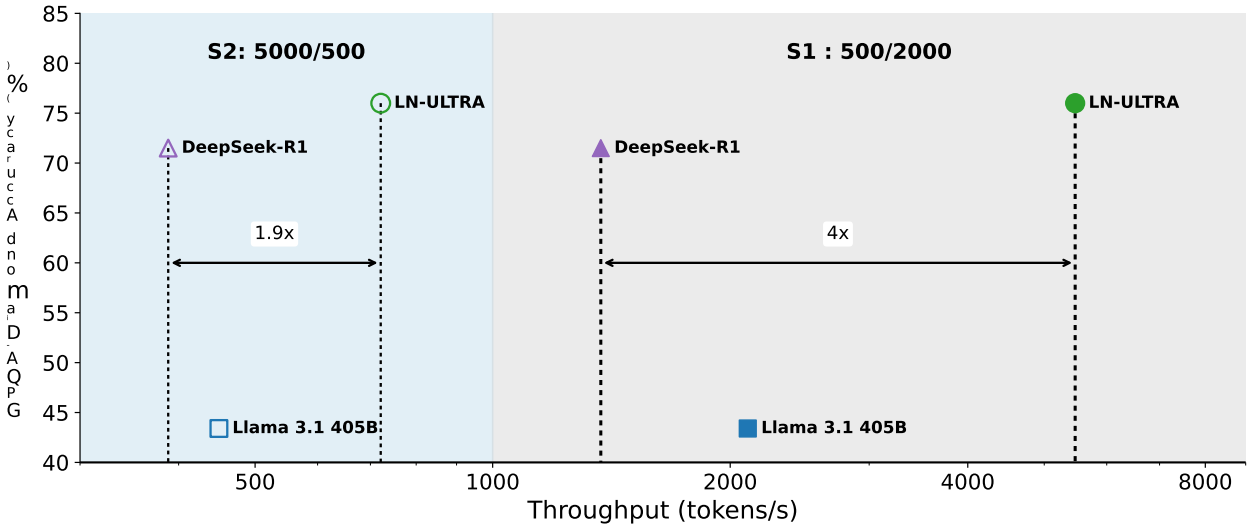


图4 | GPQA-Diamond 准确率与吞吐量。我们在两种设置下进行测量，S1：500/2000（ISL/OSL）；S2：5000/500（ISL/OSL）。两者均有250个并发用户。模型采用FP8进行服务。注意，我们使用8×H100用于LN-Ultra和Llama 3.1 405B，但由于其规模，Deepseek-R1使用8×H200。

图4展示了在两种设置下，GPQA-Diamond的准确率（%）与处理吞吐量（tokens/s）之间的权衡关系。值得注意的是，LN-Ultra在这两种设置中在准确率和效率方面始终优于DeepSeek-R1和Llama-3.1-405B，明确将其定位为在准确率-吞吐量帕累托曲线上更优的选择。

2.2. NAS 后训练：知识蒸馏与持续预训练

F在 NAS 阶段之后，LN-Super 和 LN-Ultra 都进行了额外的训练以提升性能
i块间兼容性并恢复在块式替换过程中引入的任何质量损失。

- LN-Super 使用知识蒸馏目标在 *Distillation Mix* 数据集上训练了 40B 个标记，该数据集由 Bercovich 等人（2024）引入。
- LN-Ultra 首先使用相同的蒸馏数据集进行知识蒸馏训练，覆盖 65B 个标记，然后在 Nemotron-H 第 4 阶段预训练数据集（NVIDIA 等，2025）上继续训练 88B 个标记。

这最后的预训练步骤使得LN-Ultra不仅能够与参考模型Llama 3.1-405B-Instruct相匹配，还在关键基准测试中超越它，证明了通过短期蒸馏和预训练，激进的架构优化可以与高模型性能相协调（见表1）。

Task	LN-Ultra CPT	Llama-3.3 70B-Instruct	Llama-3.1 405B-Instruct
MMLU	88.1	81.4	**88.6**
MATH500	**80.4**	73.6	69.6
HumanEval	**88.4**	84.1	86.0
RULER 128K	**83.2**	52.2	73.7

Table 1 | Comparison of LN-Ultra after the continued pretraining phase (before supervised and reinforcement learning) to Llama 3 models.

3. Synthetic Data

We curate both reasoning and non-reasoning data for supervised fine-tuning. For reasoning samples, we include the system instruction `"detailed thinking on"`, and for non-reasoning samples, we use `"detailed thinking off"`. This setup allows the model to learn to toggle reasoning behavior at inference time based on the prompt. Below, we describe our focused data curation process for each mode.

3.1. Reasoning on

3.1.1. Math

To construct the math reasoning portion of our data we used a pipeline described by Moshkov et al. (2025). A high-level overview of this pipeline is provided below, with full details available in the original publication.

We collect a large set of mathematical problems from Art of Problem Solving (AoPS) community forums. We include all forum discussions except "Middle School Math" which was found to be too easy and unhelpful for training in our early experiments. After retrieving forum discussions we perform the following steps to extract problems and synthesize new solutions. We use Qwen2.5-32B-Instruct (Qwen, 2024) for all steps in the pipeline unless noted otherwise.

Problem Extraction: We prompt an LLM to identify and extract all problems from the initial forum posts. While most posts contain a single problem, some include multiple problems or none at all.

Problem Classification: Each extracted problem is classified into the following categories:

- Proof problem or not
- Multiple choice question or not
- Binary question (yes-or-no answer) or not
- Valid problem or not. For example, problems that are lacking context or referring to other problems are considered invalid.

We remove all proof problems, multiple-choice questions, binary questions, and invalid problems from the final dataset.

Answer Extraction: We extract the final answer from forum discussions, without attempting to extract full solutions. Only the final answer expression is extracted to enable automatic correctness checking.

Benchmark Decontamination: Following Yang et al. (2023) we use an LLM-based comparison to remove questions that closely resemble those in popular math benchmarks.

Task	LN-Ultra CPT	Llama-3.3 70B-Instruct	Llama-3.1 405B-Instruct
MMLU	88.1	81.4	**88.6**
MATH500	**80.4**	73.6	69.6
HumanEval	**88.4**	84.1	86.0
RULER 128K	**83.2**	52.2	73.7

T表 1 | 继续预训练阶段后 LN-Ultra 的比较（在有监督之前）
r强化学习) 到 Llama 3 模型。

3. 合成数据

我们为有监督微调策划了推理和非推理数据。对于推理样本，我们包括系统指令 `"detailed thinking on"`，而对于非推理样本，我们使用`"detailed thinking off"`。这种设置使模型能够根据提示在推理行为之间切换。在下文中，我们将描述每种模式下的专注数据策划过程。

3.1. 推理依据

3.1.1. Math

为了构建我们的数据中的数学推理部分，我们使用了由Moshkov等人（2025年）描述的流程。以下提供了该流程的高级概述，详细信息请参见原始出版物。

我们从 Art of Problem Solving (AoPS) 社区论坛收集了大量的数学问题。我们包括所有的论坛讨论，除了“中学数学”，因为在我们的早期实验中发现它过于简单且对训练帮助不大。在检索到论坛讨论后，我们执行以下步骤以提取问题并合成新的解答。除非另有说明，否则我们在整个流程中使用 Qwen2.5-32B-Instruct (Qwen, 2024)。

P 问题提取：我们提示一个大型语言模型（LLM）识别并提取所有初始问题中的问题。
f论坛帖子。虽然大多数帖子只包含一个问题，但有些包含多个问题或没有问题。
all.
问题分类　　离子：每个提取的问题都被分类为　　以下类别：

- 证明问题还是不是
- 多项选择题或不是
- 二元问题（是或否的回答）或不是
- 有效的问题与否。例如，缺乏上下文或指涉其他问题的问题被视为无效。

我们从最终数据集中移除了所有证明题、多项选择题、二元题和无效题。

A 答案提取：我们从论坛讨论中提取最终答案，而不试图
e提取完整的解答。仅提取最终答案表达式以实现自动正确性
c见鬼。

B 基准去污：根据Yang等人（2023）的方法，我们使用基于大型语言模型（LLM）的比较
t删除与流行数学基准测试中问题非常相似的问题。

Solution generation: We prompt DeepSeek-R1 (DeepSeek-AI et al., 2025) and Qwen2.5-Math-7B-Instruct (Qwen, 2024) to solve each problem multiple times producing "reasoning" and "non-reasoning" solutions respectively. We use 16 generations per problem for DeepSeek-R1 and 64 generations per problem for Qwen2.5-Math-7B-Instruct.

Solution Filtering: As the final filtering step, we remove any solutions that do not reach the expected answer. Predicted and expected answers are compared by prompting Qwen2.5-32B-Instruct (Qwen, 2024) to judge their equivalence in the context of the problem. For problems where the final answer cannot be extracted, we treat the most common answer across all available solution candidates as the ground truth.

All prompts and scripts necessary to run the above pipeline are available in NeMo-Skills.

3.1.2. Code

The code reasoning dataset is constructed via a multi-stage process involving question collection, solution generation, and post-processing steps, as described by Ahmad et al. (2025).

Question Collection and Verification: We aggregate 28,904 unique competitive programming questions from diverse sources including TACO (Li et al., 2023), APPS (Hendrycks et al., 2021a), CodeContests (Li et al., 2022), and CodeForces (Penedo et al., 2025a), after performing exact-match deduplication. To ensure evaluation integrity against benchmarks like (Jain et al., 2025; Li et al., 2022; Chen et al., 2021; Austin et al., 2021), we rigorously check for contamination using the method from Yang et al. (2023). This involves cosine similarity checks and semantic evaluation by LLM judges (Llama-3.3-70B (Grattafiori et al., 2024), Qwen2.5-32B (Qwen, 2024)). Manual verification confirms negligible overlap ($< 0.3\%$), validating the question set.

Solution Generation: We employ DeepSeek-R1 (DeepSeek-AI et al., 2025) to generate multiple solutions per question, primarily in Python, with C++ solutions also generated for specific benchmark testing (Penedo et al., 2025b). Solutions are generated using Nucleus Sampling (Holtzman et al., 2020) (temperature 0.6, top-p 0.95) via SGLang (Zheng et al., 2024), explicitly prompting for reasoning steps enclosed in *<think>* tags.

Post-Processing and Refinement: We refine generated responses by verifying the presence of reasoning traces, extracting solution code segments (demarcated by `python...`), removing samples with code inside reasoning tags, and validating syntax using Tree Sitter (TreeSitter, 2013). This process yields approximately 488K Python samples.

Data Scaling Insights: While some studies suggest small datasets suffice for inducing reasoning (HuggingFace, 2025; Muennighoff et al., 2025; BespokeLabs, 2025; OpenThoughts, 2025), especially in mathematics, our experiments indicate large-scale data is crucial for high performance on coding benchmarks. An ablation study scaling the dataset from 25k to 736k samples showed continuous improvement. Initial scaling (25k-100k) provides gains, but focusing generation on harder problems from CodeContests before expanding to the full question set yields the most significant performance boosts. The scaling curve does not plateau, emphasizing the importance of large, diverse, and challenging problem sets for advancing code generation capabilities, suggesting a need for methods to create or source more difficult problems at scale.

3.1.3. Science

We curate a diverse set of open-ended and multiple-choice questions (MCQs) from both in-house and external sources. These include question-answer pairs extracted from StackOverflow (Stack Exchange Data, 2024) and synthetically generated MCQ questions.

解题过程：我们提示DeepSeek-R1（DeepSeek-AI 等，2025）和Qwen2.5-Math-7B-Instruct（Qwen，2024）分别多次解决每个问题，生成“推理”与“非推理”解答。我们对每个问题使用16次生成（DeepSeek-R1）和64次生成（Qwen2.5-Math-7B-Instruct）。

解决方案筛选：作为最后的筛选步骤，我们会删除任何未达到预期答案的解决方案。预测答案和预期答案通过提示Qwen2.5-32B- Instruct（Qwen，2024）进行比较，以判断它们在问题背景下的等价性。对于无法提取最终答案的问题，我们将所有可用解决方案候选中最常见的答案视为正确答案。

运行上述流程所需的所有提示和脚本都可以在 NeMo-Skills 中找到。

3.1.2. Code

T代码推理数据集通过一个多阶段的过程构建，包括问题收集，
s解决方案生成和后处理步骤，如 Ahmad 等人（2025）所述。

题目收集与验证：我们从包括TACO（Li等，2023）、APPS（Hendrycks等，2021a）、CodeContests（Li等，2022）和CodeForces（Penedo等，2025a）在内的多个来源聚合了28,904个独特的竞赛编程题目，经过精确匹配去重。为了确保对基准测试（如Jain等，2025；Li等，2022；Chen等，2021；Austin等，2021）的评估完整性，我们严格使用Yang等（2023）的方法检查污染情况。这包括余弦相似度检测和由LLM评判（Llama-3.3-70B（Grattafiori等，2024）、Qwen2.5-32B（Qwen，2024））进行的语义评估。人工验证确认重叠极少（$< 0.3\%$），验证了题目集的有效性。

解题生成：我们采用 DeepSeek-R1（DeepSeek-AI 等，2025）为每个问题生成多个解答，主要使用 Python，同时也为特定基准测试生成 C++ 解决方案（Penedo 等，2025b）。解答通过 SGLang（Zheng 等，2024）使用核采样（Holtzman 等，2020）（温度 0.6，top-p 0.95）生成，明确提示推理步骤，括在 $<think>$ 标签中。

后处理与优化：我们通过验证推理轨迹的存在、提取解题代码片段（由 `python...` 标记）、删除包含在推理标签中的代码样本，以及使用 Tree Sitter (TreeSitter, 2013) 验证语法，来优化生成的回答。此过程产生了大约 48.8 万个 Python 样本。

数据规模化洞察：虽然一些研究表明，较小的数据集足以引发推理（HuggingFace，2025；Muennighoff 等，2025；BespokeLabs，2025；OpenThoughts，2025），尤其是在数学领域，我们的实验表明，大规模数据对于在编码基准测试中实现高性能至关重要。从25k到736k样本的数据集缩放的消融研究显示持续改进。初始的缩放（25k-100k）带来了收益，但在扩展到完整题目集之前，将生成重点放在CodeContests中的难题上，能带来最显著的性能提升。缩放曲线没有达到平台，强调了大规模、多样化且具有挑战性的问题集对于推动代码生成能力的重要性，表明需要开发或获取更多难题的方法以实现大规模的难题生成。

3.1.3. Science

W策划一组多样的开放式和多项选择题（MCQs），包括内部资源
and外部资源。这些包括从StackOverflow（Stack
Exchange Data，2024）以及合成生成的多项选择题。

Synthetic Question Generation: To create synthetic questions, we define a broad set of academic topics (e.g., physics, biology, chemistry) and their subtopics using Nemotron-4-340B-Instruct (NVIDIA, 2024c). We specify multiple difficulty levels to ensure a diverse and scalable dataset. We prompt Qwen2.5 models (Qwen, 2024) to generate MCQs conditioned on the topic, subtopic, and difficulty level. Each question is verified for format compliance. Following the OpenMathInstruct-2 (Toshniwal et al., 2025) pipeline, we augment the dataset by prompting Qwen2.5 to generate variations of the original questions.

Benchmark Decontamination: To ensure fair evaluation, we perform decontamination on the entire set of questions—both real and synthetic—against the test sets of major science benchmarks such as GPQA (Rein et al., 2023), MMLU (Hendrycks et al., 2021b), and MMLU-Pro (Wang et al., 2024), following the approach outlined in Yang et al. (2023).

Solution Generation: For all questions in the dataset, we use DeepSeek-R1 (DeepSeek-AI et al., 2025) to generate multiple reasoning traces. For questions without ground-truth answers, we apply majority voting across generated solutions to infer the most likely correct answer.

3.1.4. General

For general domain data, we follow the generation pipeline established in NVIDIA (2024c). We generate synthetic prompts covering various tasks such as open QA, closed QA, extraction, and brainstorming. We also source real-world user prompts from publicly available datasets with permissive licenses. For responses, we prompt DeepSeek-R1 (DeepSeek-AI et al., 2025) for multiple generations and perform rejection sampling using the Llama-3.1-Nemotron-70B reward model (NVIDIA, 2024b). This ensures that the responses are of high quality.

3.2. Reasoning off

To train the model to follow the reasoning toggle instruction, we construct paired data where each prompt has both a reasoning response and a non-reasoning response. Specifically, we randomly sample prompts from the reasoning dataset in Section 3.1 and generate corresponding non-reasoning responses using Llama-3.1-Nemotron-70B-Instruct (NVIDIA, 2024a) for general domain prompts and Llama-3.3-70B-Instruct for others. Each response is tagged with the appropriate system instruction—`"detailed thinking on"` for reasoning and `"detailed thinking off"` for non-reasoning. This pairing enables the model to learn to modulate its reasoning behavior based on the system prompt.

Responses are then filtered according to ground truth answers or reward models. We also leverage public permissive datasets on function calling and safety, augmenting them to train the model and improve its capabilities in these areas. To further improve performance on general tasks, we use a feedback-edit system, described in Section 3.2.1.

3.2.1. General-Domain Open-ended Inference-Time Scaling

To generate high-quality general-domain open-ended responses, we employ Llama-3.1-Nemotron-70B-Instruct (NVIDIA, 2024a) in conjunction with a novel Feedback-Edit Inference-Time-Scaling system, described by Wang et al. (2025b). The process begins with 20k first-turn prompts sourced from ShareGPT (RyokoAI, 2023) and WildChat-1M (Zhao et al., 2024). We use Llama-3.1-Nemotron-70B-Instruct to generate multiple initial responses for each prompt. These responses are refined through a three-stage process: a dedicated Feedback model identifies areas for improvement, a dedicated Edit model makes targeted edits based on the feedback, and a dedicated Select model chooses the best edited response. The resulting dataset comprises 20k first-turn prompts and their corresponding high-quality responses.

合成问题生成：为了创建合成问题，我们使用Nemotron-4-340B-Instruct（NVIDIA，2024c）定义了一组广泛的学术主题（例如物理、生物、化学）及其子主题。我们指定多个难度级别，以确保数据集的多样性和可扩展性。我们提示Qwen2.5模型（Qwen，2024）根据主题、子主题和难度级别生成多项选择题（MCQs）。每个问题都经过格式符合性验证。按照OpenMathInstruct-2（Toshniwal等，2025）的流程，我们通过提示Qwen2.5生成原始问题的变体，扩充数据集。

基准去污染：为了确保公平评估，我们对所有问题集（包括真实和合成的）进行去污染，针对主要科学基准的测试集，如 GPQA（Rein 等，2023）、MMLU（Hendrycks 等，2021b）和 MMLU-Pro（Wang 等，2024），采用 Yang 等（2023）中提出的方法。

S解决方案生成：对于数据集中的所有问题，我们使用 DeepSeek-R1（DeepSeek-AI 等，2025）生成多个推理轨迹。对于没有真实答案的问题，我们应用m对生成的解进行多数投票，以推断最可能的正确答案。

3.1.4. General

对于通用领域数据，我们遵循NVIDIA（2024c）建立的生成流程。我们生成涵盖开放式问答、封闭式问答、抽取和头脑风暴等各种任务的合成提示。我们还从具有宽松许可证的公开数据集中获取真实用户提示。对于回答，我们使用DeepSeek-R1（DeepSeek-AI等，2025）进行多次生成，并利用Llama-3.1-Nemotron-70B奖励模型（NVIDIA，2024b）进行拒绝采样。这确保了回答的高质量。

3.2. 关闭推理

为了训练模型遵循推理切换指令，我们构建了配对数据，其中每个提示都包含一个推理响应和一个非推理响应。具体来说，我们从第3.1节的推理数据集中随机抽取提示，并使用Llama-3.1-Nemotron-70B-Instruct（NVIDIA，2024a）为通用领域提示生成对应的非推理响应，对于其他提示则使用Llama-3.3-70B-Instruct。每个响应都被标记上相应的系统指令——`"detailed thinking on"` 表示推理，`"detailed thinking off"` 表示非推理。这种配对方式使模型能够根据系统提示学习调节其推理行为。

然后根据真实答案或奖励模型对响应进行过滤。我们还利用关于函数调用和安全的公共宽容性数据集，进行扩充以训练模型并提升其在这些领域的能力。为了进一步提高在通用任务上的表现，我们采用了反馈编辑系统，详见第3.2.1节。

3.2.1. General-Domain Open-ended Inference-Time Scaling

为了生成高质量的通用领域开放式回答，我们采用Llama-3.1-Nemotron-70B-Instruct（NVIDIA，2024a）结合一种新颖的反馈-编辑推理时刻缩放系统，该系统由Wang等人（2025b）描述。该过程始于从ShareGPT（RyokoAI，2023）和WildChat-1M（Zhao等，2024）获取的20k第一个轮次提示。我们使用Llama-3.1-Nemotron-70B-Instruct为每个提示生成多个初始回答。这些回答通过一个三阶段的过程进行优化：专用的反馈模型识别改进空间，专用的编辑模型根据反馈进行有针对性的编辑，以及专用的选择模型选择最佳的编辑后回答。最终的数据集包括20k第一个轮次提示及其对应的高质量回答。

Domain / Split	Samples	% of total
Math	22,066,397	66.8%
Reasoning *on*	2,225,427	6.7%
Reasoning *off*	19,840,970	60.1%
Code	10,108,883	30.6%
Reasoning *on*	991,706	3.0%
Reasoning *off*	9,117,177	27.6%
Science	708,920	2.1%
Reasoning *on*	708,920	2.1%
Reasoning *off*	0	0.0%
Chat	39,792	0.12%
Reasoning *on*	8,574	0.03%
Reasoning *off*	31,218	0.09%
Instruction Following	56,339	0.17%
Safety	31,426	0.10%
Total	33,011,757	100%

Table 2 | Synthetic data by domain with reasoning splits.

4. Supervised Fine-Tuning

Supervised fine-tuning (SFT) plays a critical role in transferring reasoning capabilities into the Llama-Nemotron models. While prior stages such as NAS and CPT focus on architectural efficiency and broad knowledge transfer, SFT helps distill reasoning behavior from strong teacher models like DeepSeek-R1 (DeepSeek-AI et al., 2025) by training on task-specific reasoning traces. It also establishes fine-grained control over response style using the `"detailed thinking on/off"` instruction. Recent studies (DeepSeek-AI et al., 2025; OpenThoughts, 2025; BespokeLabs, 2025; Wen et al., 2025) have shown that this reasoning SFT can substantially improve performance on complex reasoning tasks. Our results confirm these findings, highlighting the importance of training on large-scale, high-quality reasoning traces during SFT for eliciting robust reasoning abilities in downstream usage. This section builds upon the synthetic data described in Section 3 and provides further implementation details specific to each model.

4.1. General Methodology

All models are trained using a token-level cross-entropy loss over the instruction-tuning data. For most settings, training batches mix reasoning and non-reasoning data, where prompts are paired with responses conditioned on the respective system instruction— `"detailed thinking on/off"`.

We observe that models require higher learning rates to effectively learn from long reasoning traces, especially due to sequence-length-dependent token loss averaging. Extended training over multiple epochs improves performance, particularly for smaller models, a trend also observed in prior work (Wen et al., 2025). We use Adam optimizer for training all models. Using a cosine learning rate decay with linear warmup to around 10% of total steps helps with stability of training, which was crucial for LN-Ultra.

4.2. Model-Specific Training

LN-Nano differently from other models below, undergoes a three-stage SFT pipeline with a global batch size of 256 using sequence packing with effective sequence length of 32k tokens. In the first

Domain / Split	Samples	% of total
Math	22,066,397	66.8%
Reasoning *on*	2,225,427	6.7%
Reasoning *off*	19,840,970	60.1%
Code	10,108,883	30.6%
Reasoning *on*	991,706	3.0%
Reasoning *off*	9,117,177	27.6%
Science	708,920	2.1%
Reasoning *on*	708,920	2.1%
Reasoning *off*	0	0.0%
Chat	39,792	0.12%
Reasoning *on*	8,574	0.03%
Reasoning *off*	31,218	0.09%
Instruction Following	56,339	0.17%
Safety	31,426	0.10%
Total	33,011,757	100%

表2 | 按领域划分的合成数据及推理划分。

4. 有监督的微调

监督微调（SFT）在将推理能力转移到 Llama-Nemotron 模型中起着关键作用。虽然先前的阶段如 NAS 和 CPT 侧重于架构效率和广泛的知识转移，SFT 通过在任务特定的推理轨迹上训练，有助于从强大的教师模型如 DeepSeek-R1（DeepSeek-AI 等，2025）中提炼推理行为。它还利用 `"detailed thinking on/off"` 指令建立了对响应风格的细粒度控制。最新研究（DeepSeek-AI 等，2025；OpenThoughts，2025；BespokeLabs，2025；Wen 等，2025）表明，这种推理 SFT 可以显著提升复杂推理任务的性能。我们的结果证实了这些发现，强调在 SFT 过程中使用大规模、高质量的推理轨迹进行训练的重要性，以在下游应用中激发出稳健的推理能力。本节基于第 3 节描述的合成数据，并提供了每个模型的具体实现细节。

4.1. 一般方法

A所有模型都是使用指令调优数据上的令牌级交叉熵损失进行训练的。
most 设置，训练批次混合推理和非推理数据，其中提示被配对
w第 `"detailed thinking on/off"` 个响应以各自的系统指令为条件—— `"detailed thinking on/off"`。

我们观察到模型需要更高的学习率才能有效地从长推理轨迹中学习，特别是由于序列长度依赖的标记损失平均。经过多轮扩展训练，性能得到了提升，尤其是对于较小的模型，这一趋势在之前的工作（Wen 等，2025）中也有观察到。我们使用 Adam 优化器对所有模型进行训练。采用余弦学习率衰减结合线性预热至总步骤的约 10%，有助于训练的稳定性，这对于 LN-Ultra 来说尤为关键。

4.2. 模型特定训练

LN-Nano 与下述其他模型不同，经历了一个包含三个阶段的全局 SFT 流水线
b使用批次大小为256，采用序列打包，有效序列长度为32k个标记。 在第一个

stage, the model is fine-tuned exclusively on reasoning data from code, math, and science domains (Section 3.1) with a learning rate of 1e−4 for four epochs. This prevents failure modes such as repetitive completions. In the second stage, we introduce non-reasoning data (Section 3.2) mixed with reasoning samples, allowing the model to learn reasoning control. In the final stage, a smaller blend focused on chat, instruction-following, and tool-calling is used.

LN-Super is trained on the full SFT dataset for a single epoch using a fixed learning rate of 5e−6, sequence length of 16k and a global batch size of 256. Smaller-scale runs suggest that performance improves up to 3–4 epochs with larger learning rates (5e−5), but training was constrained by computational and time limits. Recent works (Wen et al., 2025) show that rejection fine-tuning can further improve performance; however, it does not yield gains in our experiments and is therefore omitted.

LN-Ultra is trained on the full dataset using sequence packing with effective sequence length of 24k and a global batch size of 256 to maximize token throughput—an essential strategy when fine-tuning large models with long-context reasoning data. Initial ablation runs indicated that higher learning rates such as 5e−5 generally improve outcomes, but consistently high learning rates caused instability, including gradient explosions. To mitigate this, we implement a linear warmup to 1e−5, followed by cosine decay to 1e−6 with a warmup ratio of 10%. Despite these measures, training encountered gradient explosions and numerical instability after the first epoch. This required training resumption with reinitialized optimizer states, after which successful convergence was achieved.

5. RL for Reasoning

As described in Section 4 and illustrated in Table 5, models can acquire strong capabilities through supervised fine-tuning by distilling knowledge from powerful teachers. However, distillation inherently sets an upper bound on the student's performance, particularly when the student's base model is not more capable than the teacher's. Using supervised fine-tuning, LN-Ultra can approach the performance of DeepSeek-R1 but not exceed it. To enable students to surpass their teachers, large-scale reinforcement learning is a viable approach, as it allows the model to continually explore new possibilities and engage in self-learning. Consistent with the findings of DeepSeek-AI et al. (2025), our preliminary experiments indicate that applying RL to smaller models yields suboptimal results compared to distillation.Due to resource constraints, we apply reasoning RL only to LN-Ultra, which results in a model that outperforms its teacher.

5.1. Training Procedure

For LN-Ultra, we enhance the model's scientific reasoning capabilities through large-scale reinforcement learning, leveraging the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024). We use a rollout prompt size of 72 and sample 16 responses per prompt with $temperature = 1$ and $top\_p = 1$. During training, we set global batch size as 576 and conduct 2 gradient updates per rollout. We train our model until it achieves convergence on reasoning tasks. Figure 5 shows the accuracy score on GPQA-Diamond as our training progresses. With our optimized training infrastructure (see Section 5.2), the whole training takes approximately 140k H100 hours.

In this training phase, we leverage two types of rewards:

Accuracy rewards: For each training example, a ground truth answer (a number, a sentence, or a paragraph) is provided. We serve the Llama-3.3-70B-Instruct model to judge whether the policy's

阶段，模型在仅包含代码、数学和科学领域推理数据（第3.1节）上进行微调，学习率为1e−4，持续四个周期。这可以防止诸如重复完成等失败模式。在第二阶段，我们引入非推理数据（第3.2节）与推理样本混合，允许模型学习推理控制。在最后阶段，使用更小比例的混合数据，重点关注聊天、指令执行和工具调用。

LN-Super 在完整的 SFT 数据集上进行单轮训练，使用固定学习率 5e−6，序列长度为 16k，全球批次大小为 256。较小规模的实验表明，随着学习轮数增加到 3–4 轮，采用更大的学习率（5e−5）可以提升性能，但由于计算和时间限制，训练受到约束。近期的研究（Wen 等，2025）显示，拒绝微调可以进一步提升性能；然而，在我们的实验中并未取得提升，因此被省略。

LN-Ultra 使用序列打包在完整数据集上进行训练，序列长度为 24k，全球批次大小为 256，以最大化令牌吞吐量——这是微调具有长上下文推理数据的大模型时的关键策略。初步消融实验表明，较高的学习率如 5e−5 通常会改善结果，但持续的高学习率会导致不稳定，包括梯度爆炸。为此，我们采用线性预热至 1e−5，然后以余弦衰减至 1e−6，预热比例为 10%。尽管采取了这些措施，训练在第一个 epoch 后仍遇到梯度爆炸和数值不稳定的问题。这需要重新初始化优化器状态后恢复训练，之后才成功收敛。

5. 推理的强化学习

如第4节所述并在表5中展示，模型可以通过从强大的教师中蒸馏知识进行有监督的微调，获得强大的能力。然而，蒸馏本质上对学生的性能设定了一个上限，特别是当学生的基础模型不比教师更强大时。通过有监督的微调，LN-Ultra可以接近DeepSeek-R1的性能，但无法超越它。为了使学生能够超越其教师，大规模强化学习是一种可行的方法，因为它允许模型不断探索新的可能性并进行自我学习。与DeepSeek-AI等人（2025年）的研究结果一致，我们的初步实验表明，将RL应用于较小的模型，其效果不如蒸馏。由于资源限制，我们仅对LN-Ultra应用推理RL，从而得到一个性能优于其教师的模型。

5.1. 训练流程

对于 LN-Ultra，我们通过大规模强化学习提升模型的科学推理能力，采用 Group Relative Policy Optimization (GRPO) 算法（Shao 等，2024）。我们使用 72 的 rollout 提示大小，并对每个提示采样 16 个响应，使用 $temperature = 1$ 和 $top\_p = 1$。在训练过程中，我们将全局批次大小设为 576，并进行每次 rollout 2 次梯度更新。我们训练模型直到其在推理任务上收敛。图 5 显示了随着训练的进行，在 GPQA-Diamond 上的准确率得分。借助我们优化的训练基础设施（见第 5.2 节），整个训练大约耗时 14 万 H100 小时。

在这个训练阶段，我们利用两种类型的奖励：

A准确率奖励：对于每个训练样本，一个真实答案（一个数字、一句话或一个句子）
p（段落）已提供。我们使用Llama-3.3-70B-Instruct模型来判断政策是否符合。

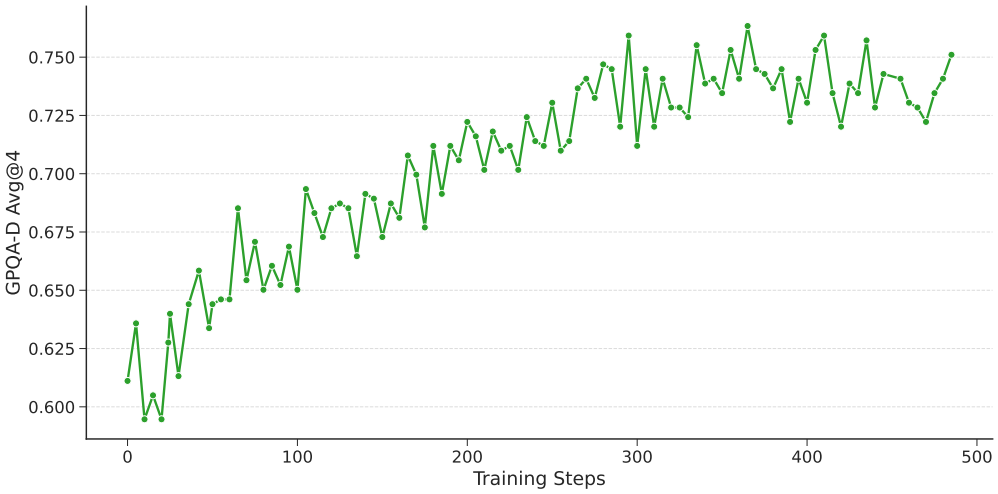


Figure 5 | Accuracy on GPQA-Diamond throughout the reasoning RL training for LN-Ultra

predictions match the ground truth answer.

Format rewards: Following DeepSeek-AI et al. (2025), we employ a format reward to ensure the model puts its thinking process between "<think>" and "</think>" tags when using `"detailed thinking on"` mode. We also check for the non-existence of thinking tags when using `"detailed thinking off"` mode.

To ensure that the model is adequately challenged, we preprocess the data by independently generating 8 responses per question using LN-Super, calculating the pass rate, and then intentionally discarding prompts with a pass rate of 0.75 or higher, thereby increasing the difficulty of the training data. Besides data filtering, we also find curriculum training to be helpful, as it allows the model to gradually learn from a progression of tasks with increasing difficulty. Specifically, we implement a progressive batching strategy leveraging pre-calculated pass rate as a difficulty metric. Given a fixed batch size, the core of our approach involves dynamically calculating a target distribution of pass rates for each sequential batch. This distribution is modeled using a Gaussian function centered on a difficulty level that progresses from high pass rates (easier examples) for initial batches to low pass rates (harder examples) for later batches. Samples are allocated to each batch primarily based on this target distribution, considering the available count for each pass rate, with any remaining batch capacity filled by prioritizing pass rates with the largest remaining sample pools. This ensures a controlled, gradual increase in average sample difficulty across batches, while samples inside a batch are randomly shuffled. Figure 6 demonstrates the effectiveness of our curriculum strategy, which stabilizes the training process and achieves higher accuracy.

5.2. Infrastructure

5.2.1. Overview

We primarily use NeMo-Aligner (Shen et al., 2024) to perform RL training, where we use a development branch that implements GRPO and heterogeneous model support. We implements the generation stage using vLLM (Kwon et al., 2023) and the training stage using Megatron-LM (Shoeybi et al., 2020). The training and inference stages are co-located on the same GPUs.

The total GPU count used was 72 nodes of 8xH100. The training model parallelism used was: tensor parallel=8 with sequence parallel, context parallel=2, pipeline parallel=18, and data parallel=2.

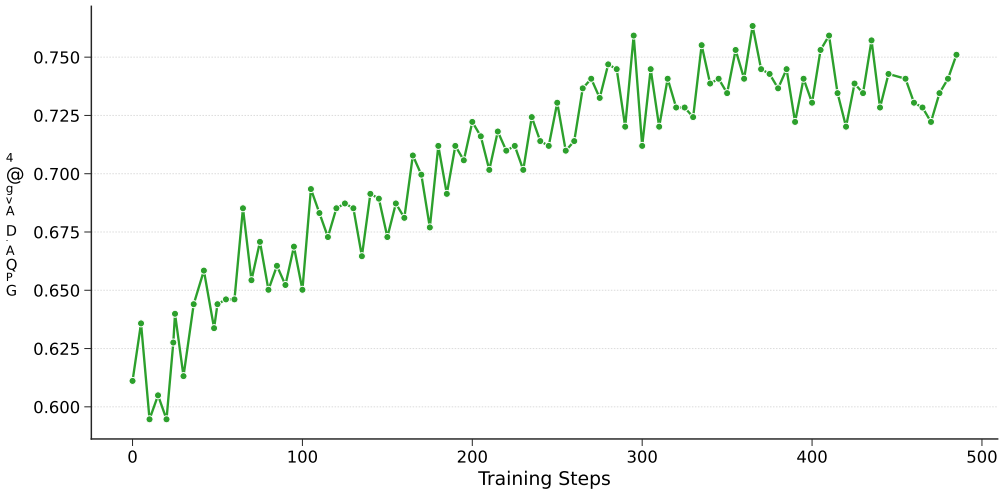


图5 | 在 LN-Ultra 的推理 RL 训练过程中，GPQA-Diamond 上的准确率

预测与真实答案相符。

格式奖励：遵循 DeepSeek-AI 等人（2025年）的做法，我们采用格式奖励，确保模型在使用 `"detailed thinking on"` 模式时，将其思考过程放在 "<think>" 和 "</think>" 标签之间。当使用 `"detailed thinking off"` 模式时，我们也会检查是否不存在思考标签。

为了确保模型受到充分挑战，我们通过独立生成每个问题的8个响应（使用LN-Super）来预处理数据，计算通过率，然后有意丢弃通过率为0.75或更高的提示，从而增加训练数据的难度。除了数据过滤外，我们还发现课程训练非常有帮助，因为它允许模型逐步学习一系列难度逐渐增加的任务。具体而言，我们实现了一种利用预先计算的通过率作为难度指标的渐进批处理策略。在固定批量大小的情况下，我们的方法核心是动态计算每个连续批次的通过率目标分布。该分布使用以一个从高通过率（较易样本）到低通过率（较难样本）逐步变化的难度水平为中心的高斯函数建模。样本的分配主要基于该目标分布，考虑每个通过率的可用样本数量，剩余的批次容量由优先考虑剩余样本池最大的通过率填充。这确保了批次之间平均样本难度的逐步控制增长，同时批次内的样本被随机打乱。图6展示了我们课程策略的有效性，它稳定了训练过程并实现了更高的准确率。

5.2. 基础设施

5.2.1. Overview

我们主要使用 NeMo-Aligner（Shen et al., 2024）进行 RL 训练，其中我们使用实现了 GRPO 和异构模型支持的开发分支。我们使用 vLLM（Kwon et al., 2023）实现生成阶段，使用 Megatron-LM（Shoeybi et al., 2020）实现训练阶段。训练和推理阶段在同一台 GPU 上共存。

T使用的GPU总数为72个节点，每个节点配备8个H100。所采用的训练模型并行方式为：tensor p并行=8 与序列并行、上下文并行=2、流水线并行=18 和数据并行=2。

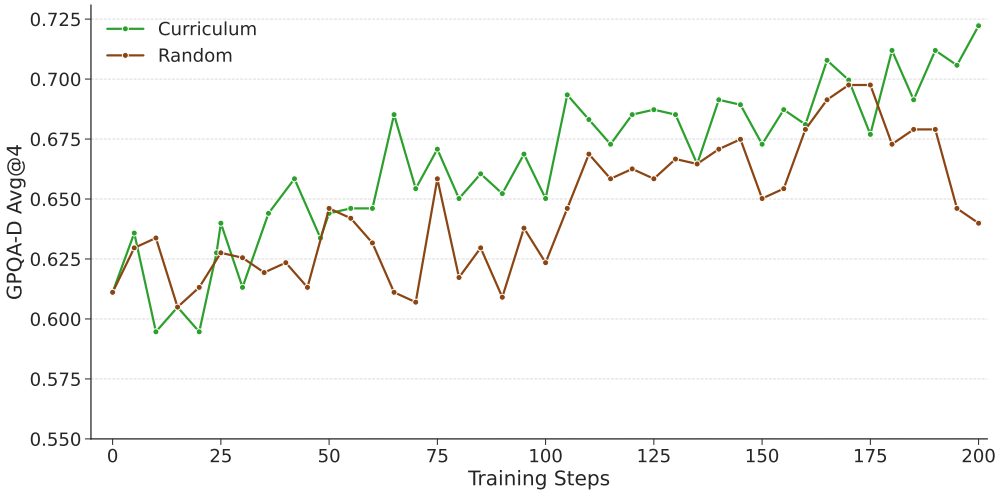


Figure 6 | Ablation on curriculum vs non-curriculum.

The generation model parallelism was tensor parallel=8, and data parallel=72. The details of how this parallelization strategy is chosen is explained in 5.2.2. Generation was performed in FP8, and training in BF16 with FP32 optimizer states.

Each stage maintains its own set of model weights, which are synced at the start of each step. First, all training weights are all-gathered over the training pipeline parallel dimension, converted into vLLM format, and written into shared memory. Then all training stage memory is released or offloaded to host. Next, vLLM is awoken from sleep mode, loads the newly saved model weights from shared memory, and begins generating. After generations have finished, vLLM GPU memory is released using sleep mode=2, and all training memory is reloaded.

5.2.2. Memory Profiling and Optimizations

One of the major challenges in enabling the GRPO training of the LN-Ultra is memory management. The training jobs are scheduled to a shared cluster environment. In the cluster, each node has 8 H100 GPUs, dual socket 32-core CPUs, and 2TB CPU DRAM. On the other hand, the model in BF16 type takes $253 \times 2 \approx 500GB$ memory. Moreover, as mentioned in Section 5.2.1, in order to improve GPU utilization, we determine to stack training and inference stages on the same set of nodes. Without careful memory management, it is very easy to encounter out-of-memory errors in both GPU and CPU memory allocations.

In order to better track the memory usage over the course of training, we have developed three simple memory profiling tools to monitor the memory usage: GPU memory utilization using PyTorch, CPU memory utilization using psutil, and */dev/shm* utilization using the df command. The GPU/CPU memory profilers help us track the GPU/CPU memory usage at different code pointers. The */dev/shm* profiler is needed as we use */dev/shm* to pass the weights from the trainer to the vLLM server, and the host is configured to allocate up to 1TB for the */dev/shm* space.

With the help of these profiling tools, we are able to pinpoint the specific memory allocations that cause out-of-memory errors, and then design solutions to overcome the issues. The first challenge is weight preparation. When we all-gather training weights across pipeline parallel stages, we have encountered extremely big tensors due to the heterogeneous architecture. One of the tensors has 13B elements, and occupies 26B GPU memory in the BF16 type. We need to release unused GPU memory periodically, and move some of the tensor conversion operations to CPU in order to control

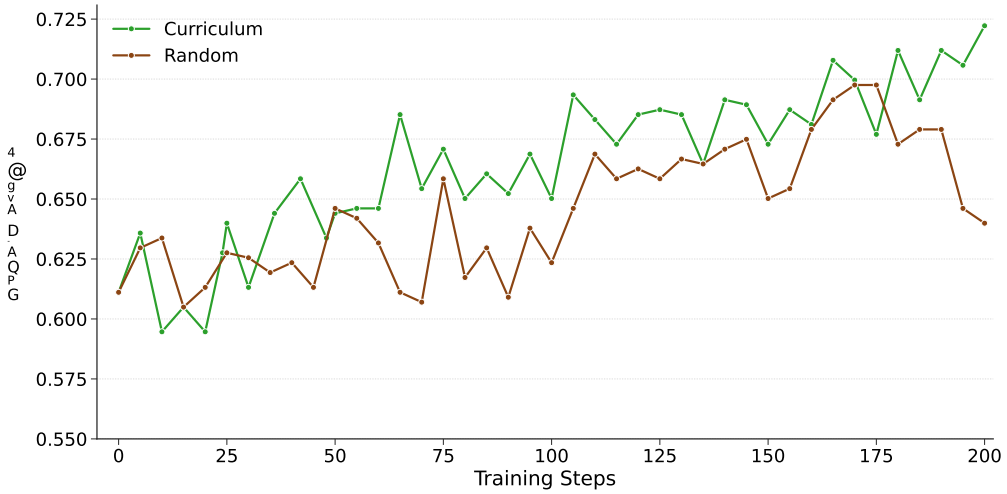


图6 | 关于课程与非课程的消融实验。

T生成模型的并行方式是张量并行=8和数据并行=72。关于具体细节，如何
t他的并行策略选择在5.2.2中进行了说明。生成是在FP8中进行的，
t在使用 FP32 优化器状态的情况下，BF16 训练中出现降雨。

每个阶段都维护自己的一套模型权重，这些权重在每个步骤开始时同步。首先，所有训练权重在训练管道并行维度上进行全收集，转换为 vLLM 格式，并写入共享内存。然后，所有训练阶段的内存被释放或卸载到主机。接下来，vLLM 从睡眠模式唤醒，从共享内存加载新保存的模型权重，并开始生成。在生成完成后，使用睡眠模式=2 释放 vLLM GPU 内存，所有训练内存被重新加载。

5.2.2. Memory Profiling and Optimizations

在实现LN-Ultra的GRPO训练时，内存管理是面临的主要挑战之一。训练任务被调度到一个共享的集群环境中。在该集群中，每个节点配备有8个H100 GPU、双插槽32核CPU以及2TB的CPU DRAM。另一方面，使用BF16类型的模型需要$253 \times 2 \approx 500GB$的内存。此外，如第5.2.1节所述，为了提高GPU的利用率，我们决定将训练和推理阶段堆叠在同一组节点上。如果没有仔细的内存管理，很容易在GPU和CPU的内存分配中遇到内存不足的错误。

为了更好地跟踪训练过程中的内存使用情况，我们开发了三种简单的内存分析工具来监控内存使用：使用 PyTorch 的 GPU 内存利用率，使用 psutil 的 CPU 内存利用率，以及使用 df 命令的 $/dev/shm$ 利用率。GPU/CPU 内存分析器帮助我们在不同的代码指针处跟踪 GPU/CPU 内存的使用情况。由于我们使用 $/dev/shm$ 将权重从训练器传递到 vLLM 服务器，并且主机配置为为 $/dev/shm$ 空间分配最多 1TB，因此需要使用 $/dev/shm$ 分析器。

在这些性能分析工具的帮助下，我们能够准确定位导致内存不足错误的具体内存分配，然后设计解决方案以克服这些问题。第一个挑战是权重准备。当我们在流水线并行的各个阶段进行训练权重的 all-gather 时，由于异构架构，我们遇到了极大的张量。其中一个张量包含 13B 个元素，在 BF16 类型下占用 26B 的 GPU 内存。我们需要定期释放未使用的 GPU 内存，并将一些张量转换操作移到 CPU，以便控制 {v*}。

the GPU memory usage in this stage. The second challenge is the vLLM GPU memory utilization. With tensor parallel equal to 8, we expect each GPU to keep $500/8 \approx 62GB$ from the weights in BF16. Considering KV cache, activations, and GPU memory occupied by the trainer, we have a very tight budget for the vLLM. We have to disable the cudagraph feature to avoid GPU out-of-memory in vLLM. However, when we enable FP8 inference generation as explained in 5.2.3, GPU memory budget becomes a lot looser, and we get to enable the cudagraph feature again. The final challenge is the GPU and CPU memory usage in the trainer. Tensor parallelism = 8 is a natural choice to partition the full model into the 8xH100 GPUs available in the same node. As the model architecture is heterogeneous, we need to insert identity layers in order to balance the pipeline stages in the pipeline parallelism. We want to have enough pipeline parallelism to avoid training OOM in GPU and checkpoint saving OOM in CPU. On the other hand, we also want to reduce the number of pipeline stages to reduce the communication costs. With all of the trade-offs, we find that the best pipeline parallelism setting is 18. The activations also consume a lot of memory, and we need to keep 18 micro-batches in the case when pipeline parallelism is 18. We end up using context parallel = 2 and sequence parallel to reduce the activation memory consumption to prevent GPU OOM in training. With all these tuning, we finally choose tensor parallel=8 with sequence parallel, context parallel=2, pipeline parallel=18, and data parallel=2 to achieve $> 90\%$ utilization of the GPUs while avoiding any hosts from encountering GPU or CPU out-of-memory errors.

5.2.3. FP8 Inference Generation

We identify the generation stage as the dominant component of the step time. In order to improve performance, we implement a path to support the use of vLLM's online FP8 generation mode, which executes all GEMMs in FP8 using per token activation scaling factors and per tensor weight scaling factors. We implement custom vLLM weight loaders capable of loading BF16 weights supplied by the training stage, and casting to FP8 weights and scaling factors at runtime. Because vLLM does not support directly initializing models in FP8, we also implement meta-weight tensor initialization to avoid materializing the full BF16 inference engine, which would cause an out-of-memory error in the GPU.

In all, we observe a peak FP8 generation throughput of 32 tokens/s/GPU/prompt, a 1.8x generation speedup against BF16, and to our knowledge the highest decoding throughput observed in reasoning training at this scale. We observe a 1.4x speedup from FP8 generation alone, and an additional 0.4x from the reduction in memory usage, which allows us to enable vLLM's cudagraph feature.

6. RL for Preference Optimization

6.1. Instruction Following

After training for scientific reasoning, we do a short RL run optimizing instruction following capabilities for the LN-Super and LN-Ultra. We use a similar verification setup as Zhou et al. (2023), and generate synthetic instruction following prompts that contain from one to ten detailed instructions. We run RL with the RLOO algorithm (Ahmadian et al., 2024) for less than 120 steps using our instruction following verifier as a reward, with a batch size of 128 prompts. We find such training boosts performance on conventional instruction following benchmarks as well as reasoning benchmarks.

6.2. RLHF

We use RLHF to improve the model on general helpfulness and chat capabilities while carefully maintaining its proficiency in other areas. As shown in Table 4, LN-Super, a 49B model, achieves an

在此阶段的GPU内存使用情况。第二个挑战是vLLM的GPU内存利用率。随着张量并行等于8，我们预计每个GPU可以保持500/8 ≈ 62GB的BF16权重。考虑到KV缓存、激活以及训练器占用的GPU内存，我们对vLLM的预算非常紧张。我们必须禁用cudagraph功能以避免vLLM中的GPU内存不足。然而，当我们启用如5.2.3节所述的FP8推理生成时，GPU内存预算变得宽松许多，我们可以再次启用cudagraph功能。最后一个挑战是训练器中的GPU和CPU内存使用情况。张量并行= 8是将完整模型划分到同一节点上的8个H100 GPU的自然选择。由于模型架构异构，我们需要插入恒等层以平衡流水线中的各个阶段。我们希望拥有足够的流水线并行性，以避免GPU训练中的OOM和CPU中断点保存的OOM。另一方面，我们也希望减少流水线阶段的数量以降低通信成本。在所有权衡中，我们发现最佳的流水线并行设置是18。激活也占用大量内存，当流水线并行为18时，我们需要保持18个微批次。最终，我们使用上下文并行= 2和序列并行，以减少激活内存消耗，防止训练中的GPU OOM。在所有这些调优之后，我们最终选择张量并行= 8，结合序列并行、上下文并行= 2、流水线并行= 18和数据并行= 2，以实现GPU利用率90%，同时避免任何主机遇到GPU或CPU内存不足的错误。

5.2.3. FP8 Inference Generation

我们将生成阶段识别为步骤时间的主要组成部分。为了提高性能，我们实现了一条支持 vLLM 在线 FP8 生成模式的路径，该模式在所有 GEMM 操作中使用 FP8 执行，采用每个 token 的激活缩放因子和每个张量的权重缩放因子。我们实现了自定义的 vLLM 权重加载器，能够加载由训练阶段提供的 BF16 权重，并在运行时将其转换为 FP8 权重和缩放因子。由于 vLLM 不支持直接在 FP8 中初始化模型，我们还实现了元权重张量初始化，以避免在 GPU 中物化完整的 BF16 推理引擎，从而防止内存溢出错误。

总的来说，我们观察到峰值FP8生成吞吐量为32个标记/秒/每GPU/提示，比BF16快1.8倍，并且据我们所知，这是在此规模推理训练中观察到的最高解码吞吐量。我们仅通过FP8生成就实现了1.4倍的加速，此外由于内存使用的减少，又带来了0.4倍的提升，这使我们能够启用vLLM的cudagraph功能。

6. 偏好优化的强化学习

6.1. 指令执行

在经过科学推理训练后，我们进行一次短暂的强化学习（RL）运行，优化LN-Super和LN-Ultra的指令遵循能力。我们采用与Zhou等人（2023）类似的验证设置，生成包含一到十个详细指令的合成指令遵循提示。我们使用RLOO算法（Ahmadian等人，2024）进行RL，步骤少于120步，利用我们的指令遵循验证器作为奖励，批次大小为128个提示。我们发现这种训练方式能提升在传统指令遵循基准以及推理基准上的表现。

6.2. RLHF

W我们使用 RLHF 来提升模型在通用帮助性和聊天能力方面的表现，同时谨慎地
m保持其在其他领域的熟练程度。如表4所示，LN-Super，一个49B模型，达到了

Arena Hard score of 88.3, beating proprietary models such as Claude 3.5 Sonnet and GPT-4o-2024-05-13 as well as much larger open models such as Llama-3.1-405b-instruct and Mistral-large-2407.

To achieve this, we use iterative online RPO (NVIDIA, 2024c; Sun et al., 2025) to maximize the reward predicted by Llama-3.1-Nemotron-70B-Reward (NVIDIA, 2024b) over prompts from HelpSteer2 (Wang et al., 2025a). For each iteration, we use a learning rate α of 4e-7, KL penalty β of 1e-5, reward scale η of 3.0, and batch size of 64, training for 500 steps. Two iterations of online RPO increase the Arena Hard score from 69.1 to 88.1. More interestingly, this process also improves the model's performance on all other adopted benchmarks except IFEval. Since neither the dataset nor the reward model is optimized for math, coding, science, or function calling, we speculate that RLHF helps the model better utilize its existing knowledge and skills.

We follow the same process for LN-Ultra, except that GRPO is employed. For each prompt, we sample 8 responses. We train the model for 30 steps, using a learning rate of 3e-7, batch size of 288, and KL penalty β of 1e-3.

For LN-Nano, we conduct two rounds of offline RPO with on-policy data. We use a mixture of reasoning and non-reasoning data with appropriate system prompts in the first round of RPO to improve reasoning control, followed by a second round with on-policy generations targeting instruction following improvements. For each RPO round we train up to 400 steps with a learning rate α of 7e-7, KL penalty β of 3e-2, and batch size of 512.

7. Evaluations on Reasoning and Chat Benchmarks

7.1. Benchmarks

We evaluate all Llama-Nemotron models across two benchmark categories: reasoning and non-reasoning.

Reasoning Benchmarks. These include the American Invitational Mathematics Examination for years 2024 (AIME24) and 2025(AIME25), GPQA-Diamond (Rein et al., 2024), LiveCodeBench (Jain et al., 2024), and MATH500 (Lightman et al., 2023). AIME25 is split into two parts: AIME25-I and AIME25-II, each containing 15 problems. For LN-Nano, we use AIME25-I only; for LN-Super and LN-Ultra, we evaluate on the full 30-question set. As AIME25 was released recently, it is less likely to overlap with training data. Thus, stronger performance on this benchmark is indicative of better generalization, especially on math problems outside the training distribution. LiveCodeBench contains questions indexed by date, and we report results on two specific ranges—(2408–2502) and (2410–2502)—to enable fair comparison with previously reported baselines.

Non-Reasoning Benchmarks. These include IFEval(Strict-Instruction) (Zhou et al., 2023) for instruction following, BFCL V2 Live (Yan et al., 2024) for tool use via function calling, and Arena-Hard (Li et al., 2024) for evaluating alignment with human conversational preferences.

All evaluations are conducted at a 32k context length, even though training was performed with a maximum sequence length of 16k for LN-Super and 24k for LN-Ultra. We observed consistent improvements when evaluating at expanded context lengths, as shorter sequence limits can truncate long reasoning traces and lead to incomplete generations—particularly on benchmarks that require multi-step reasoning. We use temperature 0.6 and top-p 0.95 for reasoning-on evaluations, and temperature 0 (greedy decoding) for reasoning-off. We generate up to 16 completions per prompt and report average pass@1 accuracy. Checkpoints are selected based on performance on a subset of reasoning benchmarks. As observed in prior works (Moshkov et al., 2025), evaluation on reasoning-heavy tasks such as AIME can exhibit high variance due to small dataset size and generation

Arena 硬分数为 88.3，超越了专有模型如 Claude 3.5 Sonnet 和 GPT-4o-2024-05-13 以及更大规模的开放模型，例如 Llama-3.1-405b-instruct 和 Mistral-large-2407。

为了实现这一目标，我们使用迭代在线RPO（NVIDIA，2024c；Sun等，2025）最大化由Llama-3.1-Nemotron-70B-Reward（NVIDIA，2024b）在HelpSteer2（Wang等，2025a）提示下预测的奖励。每次迭代，我们使用学习率α为4e-7，KL惩罚项β为1e-5，奖励缩放η为3.0，批量大小为64，训练500步。经过两次在线RPO迭代，Arena Hard得分从69.1提升到88.1。更有趣的是，这一过程还提升了模型在所有其他采用的基准测试中的表现，除了IFEval。由于数据集和奖励模型都未针对数学、编码、科学或函数调用进行优化，我们推测RLHF帮助模型更好地利用其现有的知识和技能。

W我们对LN-Ultra也采用相同的流程，只不过使用了GRPO。对于每个提示，我们s充足的8个响应。我们训练模型30个步骤，使用学习率3e-7，批量大小为288，and KL惩罚项 β 为1e-3。

对于 LN-Nano，我们进行两轮离线 RPO，使用策略内数据。在第一轮 RPO 中，我们结合推理和非推理数据，并配以适当的系统提示，以提高推理控制，然后进行第二轮，使用策略内生成内容，旨在改善指令遵循。每轮 RPO 训练最多进行 400 步，学习率为 α 的 7e-7，KL 惩罚项为 β 的 3e-2，批量大小为 512。

7. 推理与聊天基准的评估

7.1. 基准测试

W我对所有Llama-Nemotron模型在两个基准类别——推理和非推理——进行了评估。r推理。

推理基准。这些包括2024年（AIME24）和2025年（AIME25）的美国邀请数学考试（AIME）、GPQA-Diamond（Rein 等，2024）、LiveCodeBench（Jain 等，2024）以及MATH500（Lightman 等，2023）。AIME25分为两部分：AIME25-I和AIME25-II，每部分包含15个问题。对于LN-Nano，我们仅使用AIME25-I；对于LN-Super和LN-Ultra，我们在完整的30题集上进行评估。由于AIME25最近发布，较不可能与训练数据重叠。因此，在此基准上的更强表现表明更好的泛化能力，尤其是在训练分布之外的数学问题上。LiveCodeBench包含按日期索引的问题，我们报告两个特定范围的结果——(2408–2502)和(2410–2502)，以便与之前报告的基线进行公平比较。

N 关于推理基准。这些包括 IFEval(严格指令)（Zhou 等人，2023）f或指令跟随，BFCL V2 Live（Yan 等人，2024）用于通过函数调用进行工具使用，Arena-Hard（Li 等，2024）用于评估与人类会话偏好的一致性。

所有评估均在32k上下文长度下进行，尽管训练时LN-Super的最大序列长度为16k，LN-Ultra为24k。我们在扩展的上下文长度下进行评估时观察到了一致的改进，因为较短的序列限制可能会截断长的推理轨迹，导致生成不完整——尤其是在需要多步推理的基准测试中。我们在推理开启的评估中使用温度0.6和top-p 0.95，在推理关闭时使用温度0（贪婪解码）。每个提示最多生成16个完成，并报告平均的pass@1准确率。检查点的选择基于在一部分推理基准测试上的表现。正如之前的工作（Moshkov等，2025）所观察到的，像AIME这样推理密集型任务的评估可能会表现出较高的方差，这主要是由于数据集规模较小和生成的不确定性。

Task	LN-Nano-SFT Reasoning *on* | *off*	LN-Nano Reasoning *on* | *off*	DeepSeek-R1 Distilled-Llama-8B	Llama-3.1 8B-Instruct	DeepSeek-R1 Distilled Qwen-7B
GPQA-Diamond	53.5 | 33.3	**54.1** | 39.4	49.0	25.3	49.1
AIME24	**62.5** | 3.3	61.3 | 3.0	50.4	10.0	55.6
AIME25-I	51.6 | 6.6	**47.1** | 0.0	40.0	10.0	41.7
MATH500	94.4 | 38.0	**95.4** | 36.6	89.1	50.4	92.8
BFCL V2 Live	62.9 | 62.6	**63.9** | 63.6	37.8	44.3	39.2
LiveCodeBench (2408–2502)	– | –	**46.6** | –	39.6	11.8	37.6
IFEval	69.9 | 69.9	79.29 | **82.1**	73.4	81.8	67.6

Table 3 | LN-Nano and LN-Nano-SFT versus comparably sized models, split by Reasoning mode.

randomness. Reported numbers may vary across repeated runs or sampling strategies.

7.2. LN-Nano

Table 3 shows that LN-Nano achieves strong performance across all reasoning benchmarks, including AIME25-I and LiveCodeBench, despite its small size. This demonstrates the effectiveness of our SFT pipeline and curated reasoning datasets in transferring structured reasoning to compact models. For Nano reasoning SFT blends, carefully balancing data-distribution across math, coding, and stem areas has been important to achieve near state-of-the-art accuracies at SFT stage. For example, our early experiments showed worse accuracies especially in Chemistry related questions, i.e. one of the major areas in GPQA-D. Upsampling Chemistry related data samples in the STEM subset of the overall SFT blend helped to achieve higher GPQA-D accuracies. The RPO stages at the end of the post-training pipeline mainly targeted IFEval accuracy improvement as shown in Table 3.

7.3. LN-Super

Table 4 compares LN-Super to other models in its weight class. It performs competitively across both reasoning and non-reasoning tasks. In reasoning-off mode, LN-Super performs on par with Llama-3.3-70B, the model it was originally distilled from. In reasoning-on mode, it outperforms competing models such as DeepSeek-R1-Distilled-Llama-70B, providing strong reasoning capabilities without sacrificing instruction following. These results show that this single model offers the strengths of both reasoning-optimized and non-reasoning models, making it effective for general assistant and structured reasoning use cases. Additionally, as shown in Table 4, reasoning-focused SFT causes a noticeable drop in IFEval scores. To recover instruction-following capabilities, we apply a dedicated IFEval RL run (see Section 6.1) to ensure that strong reasoning does not come at the cost of degraded general assistant behavior. Our experimental results reveal another trade-off: optimizing for instruction following (as measured by IFEval) can compromise conversationality (as measured by Arena-Hard), and conversely, prioritizing conversationality may detract from instruction following performance. To address this, we applied model merging to LN-Super, selecting a checkpoint from the Pareto frontier that balances these objectives. Due to mixed outcomes, we did not adopt this approach for other models. The only area where LN-Super underperforms is on LiveCodeBench, which is attributable to its SFT phase being conducted on an earlier version of the dataset, unlike LN-Nano and LN-Ultra. We plan to address this and improve coding-related reasoning performance in a future model refresh.

Task	LN-Nano-SFT Reasoning *on* | *off*	LN-Nano Reasoning *on* | *off*	DeepSeek-R1 Distilled-Llama-8B	Llama-3.1 8B-Instruct	DeepSeek-R1 Distilled Qwen-7B
GPQA-Diamond	53.5 | 33.3	**54.1** | 39.4	49.0	25.3	49.1
AIME24	**62.5** | 3.3	61.3 | 3.0	50.4	10.0	55.6
AIME25-I	51.6 | 6.6	**47.1** | 0.0	40.0	10.0	41.7
MATH500	94.4 | 38.0	**95.4** | 36.6	89.1	50.4	92.8
BFCL V2 Live	62.9 | 62.6	**63.9** | 63.6	37.8	44.3	39.2
LiveCodeBench (2408–2502)	– | –	**46.6** | –	39.6	11.8	37.6
IFEval	69.9 | 69.9	79.29 | **82.1**	73.4	81.8	67.6

表3 | LN-Nano 和 LN-Nano-SFT 与同等规模模型的比较，按推理模式分类。

随机性。报告的数字在重复运行或采样策略中可能会有所不同。

7.2. LN-Nano

表3显示，尽管规模较小，LN-Nano在所有推理基准测试中都表现出色，包括AIME25-I和LiveCodeBench。这证明了我们的SFT流程和经过策划的推理数据集在将结构化推理迁移到紧凑模型中的有效性。对于Nano推理SFT的融合，仔细平衡数学、编码和题干领域的数据分布对于在SFT阶段实现接近最先进的准确率非常重要。例如，我们的早期实验显示，在化学相关问题上，准确率尤其较差，即GPQA-D中的主要领域之一。在整体SFT融合的STEM子集中过采样化学相关数据样本，有助于实现更高的GPQA-D准确率。后训练流程末端的RPO阶段主要针对IFEval的准确率提升，如表3所示。

7.3. LN-超級

表4将LN-Super与同类模型进行了比较。在推理和非推理任务中都表现出具有竞争力。在推理关闭模式下，LN-Super的表现与其最初蒸馏的模型Llama-3.3-70B相当。在推理开启模式下，它优于竞争模型如DeepSeek-R1-Distilled-Llama-70B，提供强大的推理能力而不牺牲指令遵循。这些结果表明，这个单一模型兼具推理优化模型和非推理模型的优势，使其在通用助手和结构化推理应用中都非常有效。此外，如表4所示，专注于推理的SFT会导致IFEval得分明显下降。为了恢复指令遵循能力，我们采用了专门的IFEval RL训练（见第6.1节），确保强大的推理能力不会以牺牲通用助手行为为代价。我们的实验结果揭示了另一个权衡：优化指令遵循（通过IFEval衡量）可能会影响对话性（通过Arena-Hard衡量），反之亦然，优先考虑对话性可能会削弱指令遵循性能。为此，我们对LN-Super进行了模型合并，选择了在帕累托前沿上平衡这些目标的检查点。由于结果不一，我们没有对其他模型采用此方法。LN-Super唯一表现不佳的领域是LiveCodeBench，这归因于其SFT阶段使用了较早版本的数据集，而非像LN-Nano和LN-Ultra那样。我们计划在未来的模型更新中解决此问题，提升编码相关推理的性能。

Task	LN-Super-SFT Reasoning *on* | *off*	LN-Super Reasoning *on* | *off*	DeepSeek-R1-Distilled-Llama-70B	QwQ-32B	Llama-3.3 70B-Instruct
GPQA-Diamond	63.8 | 46.6	**66.7** | 50.0	65.2	58.8	50.5
AIME24	63.3 | 17.5	67.5 | 16.7	70.0	**79.5**	25.8
AIME25	50.0 | 6.7	60.0 | 16.7	55.0	**65.8**	6.7
MATH500	93.2 | 76.8	**96.6** | 74.0	94.5	96.2	73.8
BFCL V2 Live	73.3 | 62.5	73.7 | **73.9**	65.5	71.6	60.4
LiveCodeBench (2408–2502)	40.9 | 28.7	45.5 | 29.7	57.5	**63.4**	-
IFEval	81.9 | 83.0	**89.2** | 89.0	85.1	86.3	92.1
Arena Hard	– | –	88.3 | –	65.4	**90.5**	72.9

Table 4 | LN-Super versus comparably sized models, split by Reasoning mode.

7.4. LN-Ultra

Table 5 and Figure 2 show that LN-Ultra matches or outperforms all existing open-weight models across reasoning and non-reasoning benchmarks. It achieves state-of-the-art performance on GPQA among open models, demonstrating the efficacy of our large-scale reinforcement learning training. Unlike prior state-of-the-art models such as DeepSeek-R1, which require 8×H200, LN-Ultra is optimized to run efficiently on a single 8×H100 node, offering improved inference throughput and deployment efficiency.

From Table 5, we observe that the LN-Ultra-SFT model approaches the performance of DeepSeek-R1 on several reasoning benchmarks, including GPQA and AIME. However, the RL stage is critical for surpassing DeepSeek-R1, particularly on GPQA. This highlights the complementary strengths of SFT and RL: SFT builds a strong foundation by distilling reasoning behavior from teacher models, while RL is essential for surpassing teacher performance and further enhancing reasoning capabilities.

We also find that there is a trade-off between the extent of SFT training and the success likelihood of subsequent RL. Although we had access to SFT checkpoints with higher benchmark scores, we initialized RL from an earlier checkpoint to improve final RL outcomes.

Task	LN-Ultra-SFT Reasoning *on* | *off*	LN-Ultra Reasoning *on* | *off*	DeepSeek R1	Llama-4 Behemoth	Llama-4 Maverick	Llama-3.1 405B-Instruct
GPQA-Diamond	66.4 | 46.0	**76.0** | 56.6	71.5	73.7	69.8	43.4
AIME24	74.6 | 46.7	**80.8** | 20.0	79.8	–	–	20.0
AIME25	60.4 | 16.7	**72.5** | 16.7	70.0	–	–	0.0
MATH500	96.6 | 84.4	97.0 | 80.4	**97.3**	95.0	–	66.2
BFCL V2 Live	74.6 | **74.9**	74.1 | 73.6	–	–	–	58.7
LiveCodeBench (2408–2502)	60.6 | 30.1	**66.3** | 29.0	65.9	–	–	–
LiveCodeBench (2410–2502)	61.8 | –	**68.1** | –	–	49.4	43.4	–
IFEval	83.2 | 79.4	88.9 | **89.5**	88.8	–	–	89.2
Arena Hard	– | –	87.0 | –	**92.0**	–	–	66.2

Table 5 | LN-Ultra versus the strongest open-weight models, split by reasoning mode.

8. Evaluations on Judging Capability

In addition to reasoning and chat capabilities where the models are trained for, we evaluate our models on an out-of-distribution task, LLM-as-a-Judge, to further assess their performance. Specifically, we

Task	LN-Super-SFT Reasoning *on* | *off*	LN-Super Reasoning *on* | *off*	DeepSeek-R1-Distilled-Llama-70B	QwQ-32B	Llama-3.3 70B-Instruct
GPQA-Diamond	63.8 | 46.6	**66.7** | 50.0	65.2	58.8	50.5
AIME24	63.3 | 17.5	67.5 | 16.7	70.0	**79.5**	25.8
AIME25	50.0 | 6.7	60.0 | 16.7	55.0	**65.8**	6.7
MATH500	93.2 | 76.8	**96.6** | 74.0	94.5	96.2	73.8
BFCL V2 Live	73.3 | 62.5	73.7 | **73.9**	65.5	71.6	60.4
LiveCodeBench (2408–2502)	40.9 | 28.7	45.5 | 29.7	57.5	**63.4**	-
IFEval	81.9 | 83.0	**89.2** | 89.0	85.1	86.3	92.1
Arena Hard	– | –	88.3 | –	65.4	**90.5**	72.9

表4 | LN-Super 与同等规模模型的比较，按推理模式分类。

7.4. LN-Ultra

表5和图2显示，LN-Ultra在推理和非推理基准测试中都能匹配或超越所有现有的开源模型。在GPQA上，它在开源模型中实现了最先进的性能，展示了我们大规模强化学习训练的有效性。与之前的最先进模型如DeepSeek-R1（需要8×H200）不同，LN-Ultra经过优化，可以在单个8×H100节点上高效运行，提供更高的推理吞吐量和部署效率。

从表5可以看出，LN-Ultra-SFT模型在多个推理基准测试中接近DeepSeek-R1的性能，包括GPQA和AIME。然而，RL阶段对于超越DeepSeek-R1至关重要，尤其是在GPQA上。这突显了SFT和RL的互补优势：SFT通过从教师模型中提炼推理行为建立坚实的基础，而RL对于超越教师性能和进一步提升推理能力至关重要。

W我还发现，SFT训练的程度与成功的可能性之间存在权衡
of 后续的强化学习。虽然我们可以使用具有更高基准分数的SFT检查点，然而我们
i用早期的检查点初始化RL，以改善最终的RL结果。

Task	LN-Ultra-SFT Reasoning *on* | *off*	LN-Ultra Reasoning *on* | *off*	DeepSeek R1	Llama-4 Behemoth	Llama-4 Maverick	Llama-3.1 405B-Instruct
GPQA-Diamond	66.4 | 46.0	**76.0** | 56.6	71.5	73.7	69.8	43.4
AIME24	74.6 | 46.7	**80.8** | 20.0	79.8	–	–	20.0
AIME25	60.4 | 16.7	**72.5** | 16.7	70.0	–	–	0.0
MATH500	96.6 | 84.4	97.0 | 80.4	**97.3**	95.0	–	66.2
BFCL V2 Live	74.6 | **74.9**	74.1 | 73.6	–	–	–	58.7
LiveCodeBench (2408–2502)	60.6 | 30.1	**66.3** | 29.0	65.9	–	–	–
LiveCodeBench (2410–2502)	61.8 | –	**68.1** | –	–	49.4	43.4	–
IFEval	83.2 | 79.4	88.9 | **89.5**	88.8	–	–	89.2
Arena Hard	– | –	87.0 | –	**92.0**	–	–	66.2

表5 | LN-Ultra 与最强的开源模型在推理模式上的对比。

8. 评估判断能力

我除了模型训练的推理和聊天能力之外，我们还评估了我们的模型
o在一个分布外任务中，LLM作为评判者，以进一步评估它们的性能。具体来说，我们

test them on JudgeBench (Tan et al., 2025), where the task is to differentiate between high-quality and low-quality responses. As shown in Table 6, our models outperform top proprietary and open-source models. Notably, LN-Ultra emerges as the best open-source model, significantly surpassing DeepSeek-R1 and trailing only behind o3-mini(high). Furthermore, LN-Super also outperforms o1-mini, demonstrating that our models exhibit strong generalization capabilities across diverse tasks.

Model	Knowledge	Reasoning	Math	Coding	Overall
o1-preview	66.23	79.59	85.71	85.71	75.43
o1-mini	58.44	62.24	82.14	78.57	65.71
o3-mini(low)	62.99	69.39	83.93	83.33	70.57
o3-mini(medium)	62.34	86.73	85.71	92.86	76.57
o3-mini(high)	67.53	**89.80**	87.50	**100.0**	**80.86**
DeepSeek-R1	59.09	82.65	80.36	92.86	73.14
LN-Super	64.94	67.35	76.79	83.33	69.71
LN-Ultra	**70.13**	81.63	**89.29**	92.86	79.14

Table 6 | Llama-Nemotron models demonstrate strong performance on JudgeBench.

9. Conclusions

We present the Llama-Nemotron series of models. The models are released under a permissive license and we open-source the weights, training data, and code. The Llama-Nemotron series of models perform competitively with state-of-the-art reasoning models, while having low memory requirements and efficient inference capabilities.

We find that in the presence of a strong reasoning teacher, supervised fine-tuning on high-quality synthetic data generated by such teacher is very effective in adding reasoning capabilities to smaller models. However, to push reasoning capabilities beyond what is possible from a teacher reasoning model alone, it is necessary to run large-scale, curriculum-driven reinforcement learning from verifiable rewards training.

We also show that to produce a great all-around model, e.g. a model which performs well on a wide variety of benchmarks, it is necessary to have several stages in the post-training pipeline.

Contributors

We thank the following people for their invaluable contributions to the Llama-Nemotron effort.

NAS, Distillation and Continued Pretraining

Akhiad Bercovich*, Itay Levy*, Izik Golan*, Mohammad Dabbah*, Ran El-Yaniv*, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein

Post-training

Jiaqi Zeng*, Soumye Singhal*, Alexander Bukharin*, Yian Zhang*, Tugrul Konuk*, Gerald Shen*, Ameya Sunil Mahabaleshwarkar*, Bilal Kartal*, Yoshi Suhara*, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia

在 JudgeBench (Tan 等人，2025) 上测试它们，任务是区分高质量和低质量的回答。如表6所示，我们的模型优于顶级的专有和开源模型。值得注意的是，LN-Ultra 成为表现最好的开源模型，显著超越 DeepSeek-R1，仅次于 o3-mini(high)。此外，LN-Super 也优于 o1-mini，表明我们的模型在各种任务中展现出强大的泛化能力。

Model	Knowledge	Reasoning	Math	Coding	Overall
o1-preview	66.23	79.59	85.71	85.71	75.43
o1-mini	58.44	62.24	82.14	78.57	65.71
o3-mini(low)	62.99	69.39	83.93	83.33	70.57
o3-mini(medium)	62.34	86.73	85.71	92.86	76.57
o3-mini(high)	67.53	**89.80**	87.50	**100.0**	**80.86**
DeepSeek-R1	59.09	82.65	80.36	92.86	73.14
LN-Super	64.94	67.35	76.79	83.33	69.71
LN-Ultra	**70.13**	81.63	**89.29**	92.86	79.14

表6 | Llama-Nemotron模型在JudgeBench上表现出色。

9. 结论

我们推出了Llama-Nemotron系列模型。这些模型在宽松的许可证下发布，我们开源了权重、训练数据和代码。Llama-Nemotron系列模型在推理能力方面与最先进的模型具有竞争力，同时具有低内存需求和高效的推理能力。

我们发现，在有强大推理教师的情况下，在由此类教师生成的高质量合成数据上进行监督微调，对于增强较小模型的推理能力非常有效。然而，为了将推理能力提升到仅靠教师推理模型所能达到的水平之外，有必要进行大规模、课程驱动的基于可验证奖励的强化学习训练。

We 也显示，为了产生一个全面优异的模型，例如在各种任务中表现良好的模型，
v在各种基准测试中，后训练流程中需要经过多个阶段。

贡献者

我们感谢以下人员对Llama-Nemotron项目做出的宝贵贡献。

NAS、蒸馏与持续预训练

阿希亚德·贝尔科维奇*，伊泰·莱维*，伊齐克·戈兰*，穆罕默德·达巴*，兰·埃尔-雅尼夫*，奥姆里·普尼，伊多·加利尔，扎克·莫谢，托默·罗嫩，纳吉布·纳布瓦尼，伊多·沙哈夫，奥伦·特罗普，埃胡德·卡帕斯，兰·齐尔伯斯坦

后训练

Jiaqi Zeng*、Soumye Singhal*、Alexander Bukharin*、Yian Zhang*、Tugrul Konuk*、Gerald Shen*、Ameiya Sunil Mahabaleshwarkar*，Bilal Kartal*，Yoshi Suhara*，Olivier Delalleau，Zijia Chen，Zhilin W安格，David Mosallanezhad，Adi Renduchintala，Qian Haifeng，Dima Rekesh，Fei Jia

Data

Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jiaqi Zeng, Zhilin Wang, Tugrul Konuk, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe

Infrastructure

Jimmy Zhang*, Bor-Yiing Su*, Guyue Huang*, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen

Inference

Izzy Putterman*, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw

Evaluations and Safety

Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Soumye Singhal*

Program management

Joey Conway*, Ehud Karpas*, Trisha Saar*, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga

Leadership

Tugrul Konuk*, Oleksii Kuchaiev*, Boris Ginsburg*, Oluwatobi Olabiyi*, Kari Briski*, Jonathan Cohen*, Bryan Catanzaro*, Jonah Alben, Yonatan Geifman, Eric Chung

* Core contributor

数据

Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, Georg e Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakh-turina, Jiaqi Zeng, Zhilin Wang, Tugrul Konuk, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Jane Polak Scowcroft, J ohn Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder P armar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mo stofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe

基础设施

Jimmy Zhang*、Bor-Yiing Su*、Guyue Huang*、Terry Kong、Parth Chadha、Sahil Jain、Christine H哈维、埃拉德·塞加尔、黄济宁、谢尔盖·卡希尔斯基、罗伯特·麦奎恩

推理

Izzy Putterman*、George Lam、Arun Venkatesan、Sherry Wu、Vinh Nguyen、Manoj Kilaru、Andrew Wang、Anna Warno、Abhilash Somasamudramath、Sandip Bhaskar、Maka Dong、Nave Assaf、Shaha r Mor、Omer Ullman Argov、Scot Junkin、Oleksandr Romanenko、Pedro Larroy、Monika Katariya、Marco Rovinelli、Viji Balas、Nicholas Edelman、Anahita Bhiwandiwalla、Muthu Subramaniam、Smita Ithape、Karthik Ramamoorthy、Yuting Wu、Suguna Varshini Velury、Omri Almog、Joyjit Daw

评估与安全

D恩尼斯·弗里德曼，埃里克·加林根，迈克尔·埃文斯，凯瑟琳·卢纳，莱昂·德尔克斯基，妮基·波普，E伊琳·朗、塞斯·施耐德、吉列尔莫·西曼、托马什·格热戈泽克、巴勃罗·里巴尔塔、莫妮卡·卡塔里亚，Soumye Singhal*

项目管理

乔伊·康威*，埃胡德·卡帕斯*，特丽莎·萨尔*，关安，克日什托夫·帕韦莱克，夏玛拉·普拉雅加

领导力

Tugrul Konuk*, Oleksii Kuchaiev*, Boris Ginsburg*, Oluwatobi Olabiyi*, Kari Briski*, Jonathan Cohen*, Bryan Catanzaro*, Jonah Alben, Yonatan Geifman, Eric Chung

* 核心贡献者

References

Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. OpenCodeReasoning: Advancing Data Distillation for Competitive Coding, 2025. URL https://arxiv.org/abs/2504.01943.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.298. URL https://doi.org/10.18653/v1/2023.emnlp-main.298.

Anthropic. Claude 3.7 sonnet and claude code. https://www.anthropic.com/news/claude-3-7-sonnet, February 2025. Accessed: 2025-04-26.

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.

Akhiad Bercovich, Tomer Ronen, Talor Abramovich, Nir Ailon, Nave Assaf, Mohammad Dabbah, Ido Galil, Amnon Geifman, Yonatan Geifman, Izhak Golan, Netanel Haber, Ehud Karpas, Roi Koren, Itay Levy, Pavlo Molchanov, Shahar Mor, Zach Moshe, Najeeb Nabwani, Omri Puny, Ran Rubin, Itamar Schen, Ido Shahaf, Oren Tropp, Omer Ullman Argov, Ran Zilberstein, and Ran El-Yaniv. Puzzle: Distillation-Based NAS for Inference-Optimized LLMs, 2024. URL https://arxiv.org/abs/2411.19146.

Akhiad Bercovich, Mohammad Dabbah, Omri Puny, Ido Galil, Amnon Geifman, Yonatan Geifman, Izhak Golan, Ehud Karpas, Itay Levy, Zach Moshe, Najeeb Nabwani, Tomer Ronen, Itamar Schen, Elad Segal, Ido Shahaf, Oren Tropp, Ran Zilberstein, and Ran El-Yaniv. Ffn fusion: Rethinking sequential computation in large language models, 2025. URL https://arxiv.org/abs/2503.18908.

BespokeLabs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2025-01-22.

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.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong,

参考文献

Wasi Uddin Ahmad、Sean Narenthiran、Somshubra Majumdar、Aleksander Ficek、Siddhartha Jain、Jocelyn Huang、Vahid Noroozi 和 Boris Ginsburg。OpenCodeReasoning：推动数据蒸馏以提升竞争性编码，2025。网址 https://arxiv.org/abs/2504.01943。Arash Ahmadian、Chris Cremer、Matthias Gallé、Marzieh Fadaee、Julia Kreutzer、Olivier Pietquin、Ahmet Üstün 和 Sara Hooker。回归基础：重新审视强化风格优化以从人类反馈中学习在 llms 中的应用。*arXiv preprint arXiv:2402.14740*，2024。Joshua Ainslie、James Lee-Thorp、Michiel de Jong、Yury Zemlyanskiy、Federico Lebrón 和 Sumit Sanghai。GQA：从多头检查点训练通用多查询变换器模型。在 Houda Bouamor、Juan Pino 和 Kalika Bali（编辑）中，*Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*，第 4895–4901 页。计算语言学协会，2023。doi：10.18653/V1/2023.EMNLP-MAIN.298。网址 https://doi.org/10.18653/v1/2023.emnlp-main.298。Anthropic。Claude 3.7 Sonnets 和 Claude 代码。https://www.anthropic.com/news/ claude-3-7-sonnet，2025年2月。访问时间：2025-04-26。Jacob Austin、Augustus Odena、Maxwell Nye、Maarten Bosma、Henryk Michalewski、David Dohan、Ellen Jiang、Carrie Cai、Michael Terry、Quoc Le 和 Charles Sutton。使用大型语言模型进行程序合成，2021。网址 https://arxiv.org/abs/2108.07732。Akhiad Bercovich、Tomer Ronen、Talor Abramovich、Nir Ailon、Nave Assaf、Mohammad Dabbah、Ido Galil、Amnon Geifman、Yonatan Geifman、Izhak Golan、Netanel Haber、Ehud Karpas、Roi Koren、Itay Levy、Pavlo Molchanov、Shahar Mor、Zach Moshe、Najeeb Nabwani、Omri Puny、Ran Rubin、Itamar Schen、Ido Shahaf、Oren Tropp、Omer Ullman Argov、Ran Zilberstein 和 Ran El-Yaniv。谜题：基于蒸馏的 NAS 用于推理优化的 LLMs，2024。网址 https://arxiv.org/abs/2411.19146。Akhiad Bercovich、Mohammad Dabbah、Omri Puny、Ido Galil、Amnon Geifman、Yonatan Geifman、Izhak Golan、Ehud Karpas、Itay Levy、Zach Moshe、Najeeb Nabwani、Tomer Ronen、Itamar Schen、Elad Segal、Ido Shahaf、Oren Tropp、Ran Zilberstein 和 Ran El-Yaniv。FFN 融合：重新思考大型语言模型中的顺序计算，2025。网址 https://arxiv.org/abs/2503.18908。BespokeLabs。Bespoke-stratos：推理蒸馏的非理性有效性。www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation，2025。访问时间：2025-01-22。Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde De Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman 等。评估在代码上训练的大型语言模型。*arXiv preprint arXiv:2107.03374*，2021。DeepSeek-AI、Daya Guo、Dejian Yang、Haowei Zhang、Junxiao Song、Ruoyu Zhang、Runxin Xu、Qihao Zhu、Shirong Ma、Peiyi Wang、Xiao Bi、Xiaokang Zhang、Xingkai Yu、Yu Wu、Z. F. Wu、Zhibin Gou、Zhihong Shao、Zhuoshu Li、Ziyi Gao、Aixin Liu、Bing Xue、Bingxuan Wang、Bochao Wu、Bei Feng、Chengda Lu、Chenggang Zhao、Chengqi Deng、Chenyu Zhang、Chong Ruan、Damai Dai、Deli Chen、Dongjie Ji、Erhang Li、Fangyun Lin、Fucong Dai、Fuli Luo、Guangbo Hao、Guanting Chen、Guowei Li、H. Zhang、Han Bao、Hanwei Xu、Haocheng Wang、Honghui Ding、Huajian Xin、Huazuo Gao、Hui Qu、Hui Li、Jianzhong Guo、Jiashi Li、Jiawei Wang、Jingchang Chen、Jingyang Yuan、Junjie Qiu、Junlong Li、J. L. Cai、Jiaqi Ni、Jian Liang、Jin Chen、Kai Dong,

Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Wang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2501.12948.

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

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring Coding Challenge Competence With APPS. *NeurIPS*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding, 2021b. URL https://arxiv.org/abs/2009.03300.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

HuggingFace. Open R1: A fully open reproduction of DeepSeek-R1, January 2025. URL https://github.com/huggingface/open-r1.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

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

胡凯，高凯歌，关康，黄克新，库快，于快，王 Lean，张乐聪，赵亮，王立通，张丽月，徐磊，夏乐怡，张明川，张明华，汤明慧，李孟，王苗俊，李明明，田宁，黄盼盼，张鹏，王千成，陈琼瑜，杜秋雨，葛瑞奇，王瑞松，潘瑞哲，王润基，陈R. J.，金R. L.，陈如意，陆尚豪，周尚燕，陈善煌，叶胜峰，王诗雨，余水平，周顺峰，潘舒婷，李S. S.，周双，吴少庆，叶胜峰，云涛，孙天佩，孙天宇，T.王，曾旺丁，赵万佳，刘文，梁文峰，高文俊，余文钦，张文涛，W. L.肖，安伟，刘晓东，王晓涵，陈晓康，聂晓涛，程新，刘新，谢新，刘兴超，杨新瑜，李新远，苏雪成，林旭恒，李X. Q.，金向越，沈晓金，陈晓莎，孙晓文，王晓翔，宋新楠，周欣怡，王先祖，山夏山，李Y. K.，王Y. Q.，魏Y. X.，张杨，徐艳红，李耀，赵耀，孙耀峰，王耀辉，余一，张一超，马一豪，刘一远，郭永强，欧元，王玉端，龚岳，邹宇恒，何雨佳，熊云帆，罗宇翔，尤宇翔，刘宇轩，周宇阳，朱一X.，徐艳红，黄雅萍，李耀辉，郑怡，朱宇辰，马云贤，汤颖，赵昕，闫玉婷，Ren Zehui，任泽辉，沙张莉，傅哲，徐哲，谢振达，张振岩，郝志成，严志刚，吴志宇，顾子慧，朱子佳，刘子俊，李子维，谢子伟，宋子阳，潘子正，黄振，许志鹏，张忠宇，张振，张震。DeepSeek-R1：通过强化学习激励大规模语言模型的推理能力，2025年。网址 https://arxiv.org/abs/2501.12948.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan 等。Llama 3 模型群。*arXiv preprint arXiv:2407.21783*，2024年。

D安·亨德里克斯、史蒂文·巴萨特、索拉夫·卡达瓦斯、马坦塔·马泽伊卡、阿库尔·阿罗拉、伊桑·郭、科林·伯恩斯、萨米尔·普拉尼克、霍拉斯·何、唐·宋和雅各布·斯坦哈特。用APPS衡量编码挑战能力。*NeurIPS*，2021a。

丹·亨德里克斯、科林·伯恩斯、史蒂文·巴萨特、安迪·邹、马纳塔斯·马泽伊卡、唐·宋和雅各布·斯坦哈特。衡量大规模多任务语言理解，2021b。网址 https://arxiv.org/abs/2009.03300。

A里·霍尔茨曼、简·布伊斯、杜力、麦克斯韦·福尔布斯和崔贞恩。神经文本退化的奇异案例。发表于 *International Conference on Learning Representations*，2020。网址 https://openreview.net/forum?id=rygGQyrFvH。

HUggingFace。Open R1：DeepSeek-R1的完全开源复刻版，2025年1月。网址 https://github.com/huggingface/open-r1。

纳曼·贾因、韩王、亚历克斯·古、李文丁、闫凡佳、张天俊、王思达、阿曼多·索拉尔-莱萨玛、森康希克、斯托伊卡。Livecodebench：对大型语言模型进行全面且无污染的代码评估。*arXiv preprint arXiv:2403.07974*，2024年。

纳曼·贾因、韩王、阿列克斯·古、李文丁、闫凡佳、张天俊、王思达、阿曼多·索拉尔-莱萨玛、科希克·森、伊恩·斯托伊卡。Livecodebench：对大型代码语言模型的整体且无污染的评估。在 *The Thirteenth International Conference on Learning Representations*，2025年。网址 https://openreview.net/forum?id=chfJJYC3iL。

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, 和 Ion Stoica。 大型语言模型的高效内存管理

Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in Algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From Live Data to High-Quality Benchmarks: The Arena-Hard Pipeline, April 2024. URL https://lmsys.org/blog/2024-04-19-arena-hard/.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-Level Code Generation with AlphaCode. *arXiv preprint arXiv:2203.07814*, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*, 2023.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 Winning Solution: Building State-of-the-Art Mathematical Reasoning Models with OpenMathReasoning dataset, 2025. URL https://arxiv.org/abs/2504.16891.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

NVIDIA. Llama-3.1-nemotron-70b-instruct. https://huggingface.co/nvidia/Llama-3.1-Nemotron-70B-Instruct, 2024a.

NVIDIA. Llama-3.1-nemotron-70b-reward. https://huggingface.co/nvidia/Llama-3.1-Nemotron-70B-Reward-HF, 2024b.

NVIDIA. Nemotron-4 340B Technical Report, 2024c. URL https://arxiv.org/abs/2406.11704.

NVIDIA, Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, Andrew Tao, Anna Shors, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Bobby Chen, Boris Ginsburg, Boxin Wang, Brandon Norick, Brian Butterfield, Bryan Catanzaro, Carlo del Mundo, Chengyu Dong, Christine Harvey, Christopher Parisien, Dan Su, Daniel Korzekwa, Danny Yin, Daria Gitman, David Mosallanezhad, Deepak Narayanan, Denys Fridman, Dima Rekesh, Ding Ma, Dmytro Pykhtar, Dong Ahn, Duncan Riach, Dusan Stosic, Eileen Long, Elad Segal, Ellie Evans, Eric Chung, Erick Galinkin, Evelina Bakhturina, Ewa Dobrowolska, Fei Jia, Fuxiao Liu, Gargi Prasad, Gerald Shen, Guilin Liu, Guo Chen, Haifeng Qian, Helen Ngo, Hongbin Liu, Hui Li, Igor Gitman, Ilia Karmanov, Ivan Moshkov, Izik Golan, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jarno Seppanen, Jason Lu, Jason Sewall, Jiaqi Zeng, Jiaxuan You, Jimmy Zhang, Jing Zhang, Jining Huang, Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jon Barker, Jonathan Cohen, Joseph Jennings, Jupinder Parmar, Karan Sapra, Kari Briski, Kateryna Chumachenko, Katherine Luna, Keshav

使用PagedAttention进行服务。在*Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*，2023年。李荣奥、傅杰、张博文、黄涛、孙志红、吕辰、刘广、金志、李戈。TACO：算法代码生成主题数据集。*arXiv preprint arXiv:2312.14852*，2023年。李天乐、姜伟林、埃文·弗里克、丽莎·邓拉普、朱邦华、何塞·E·冈萨雷斯、伊恩·斯托伊卡。从实时数据到高质量基准：Arena-Hard管道，2024年4月。网址https://lmsys.org/blog/2024-04-19-arena-hard/。李玉佳、崔大卫、郑俊英、纳特·库什曼、朱利安·施普特维瑟、雷米·勒布隆、汤姆·埃克尔斯、詹姆斯·基林、费利克斯·吉梅诺、阿古斯丁·达尔·拉戈、托马斯·休伯特、彼得·乔伊、赛普里安·德·马索恩·达乌姆、伊戈尔·巴布什金、陈新云、黄波森、约翰内斯·韦布尔、斯文·戈瓦尔、阿列克谢·切列帕诺夫、詹姆斯·莫洛伊、丹尼尔·曼科维茨、埃斯米·萨瑟兰·罗布森、普什米特·科利、南多·德弗雷塔斯、科拉伊·卡武库奥格鲁、奥里奥尔·维尼亚尔。使用AlphaCode进行竞赛级代码生成。*arXiv preprint arXiv:2203.07814*，2022年。亨特·莱特曼、维尼特·科萨拉朱、尤拉·布尔达、哈里·爱德华兹、鲍文·贝克、泰迪·李、雅恩·莱克、约翰·舒尔曼、伊利亚·苏茨克弗、卡尔·科贝。让我们一步步验证。*arXiv preprint arXiv:2305.20050*，2023年。伊万·莫什科夫、达拉·汉利、伊万·索罗金、舒布哈姆·托什尼瓦尔、克里斯托夫·亨克尔、贝尼迪克特·希弗勒、魏都、伊戈尔·吉特曼。AIMO-2获胜方案：构建最先进的数学推理模型，使用OpenMathReasoning数据集，2025年。网址https://arxiv. org/abs/2504.16891。尼克拉斯·穆恩尼霍夫、子通·杨、魏佳·史、李香、李飞飞、哈贾希尔兹、卢克·泽特勒莫耶、珀西·梁、埃马纽埃尔·坎德斯、哈希莫托·塔次诺里。s1：简单的测试时缩放，2025年。网址https://arxiv.org/abs/2501.19393。NVIDIA。Llama-3.1-nemotron-70b-instruct。https://huggingface.co/nvidia/Llama-3. 1-Nemotron-70B-Instruct，2024年。NVIDIA。Llama-3.1-nemotron-70b-reward。https://huggingface.co/nvidia/Llama-3. 1-Nemotron-70B-Reward-HF，2024年。NVIDIA。Nemotron-4 340B技术报告，2024年。网址https://arxiv.org/abs/2406.11704。NVIDIA、亚伦·布莱克曼、阿尔蒂·巴桑特、阿布欣·卡塔尔、阿迪西亚·伦杜钦塔拉、阿希亚德·贝尔科维奇、亚历山大·菲策克、阿莱克西斯·比约林、阿里·塔吉巴赫希、阿马拉·桑贾伊·德什穆克、阿梅雅·苏尼尔·马哈巴莱什瓦尔、安德鲁·陶、安娜·肖尔斯、阿什瓦斯·艾塔尔、阿什温·普贾里、阿育·达塔古普塔、巴拉拉姆·布达拉朱、鲍比·陈、鲍里斯·金斯堡、沃辛·王、布兰登·诺里克、布莱恩·巴特菲尔德、布莱恩·卡坦扎罗、卡洛·德尔·穆恩多、程瑜东、克里斯汀·哈维、克里斯托弗·帕里西恩、丹·苏、丹尼尔·科泽克瓦、丹尼·尹、达里亚·吉特曼、戴维·莫萨拉内扎德、迪帕克·纳拉扬南、德尼斯·弗里德曼、迪马·雷克什、丁马、迪米特里·皮克塔尔、东安、邓肯·里亚赫、杜桑·斯托西奇、艾琳·龙、埃拉德·塞加尔、艾莉·埃文斯、埃里克·钟、埃里克·加林根、伊夫琳·巴赫图琳娜、艾娃·多布罗沃尔斯卡、费伊·贾、福晓·刘、甘吉·普拉萨德、杰拉尔德·沈、桂林·刘、郭晨、海峰·钱、海伦·吴、洪斌·刘、惠莉、伊戈尔·吉特曼、伊利亚·卡尔马诺夫、伊万·莫什科夫、伊兹克·戈兰、雅恩·考茨、简·波拉克·斯考克罗夫特、贾里德·卡斯珀、雅尔诺·塞帕宁、杰森·卢、杰森·西沃尔、曾佳琦、尤佳轩、张静、张靖、黄金泽、薛金泽、乔斯林·黄、乔伊·康威、约翰·卡马卢、乔恩·巴克、乔纳森·科恩、约瑟夫·詹宁斯、朱平德·帕尔马、卡兰·萨普拉、卡莉·布里斯基、卡特琳娜·丘马琴科、凯瑟琳·卢纳、科什瓦

Santhanam, Kezhi Kong, Kirthi Sivamani, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Lawrence McAfee, Leon Derczynski, Lindsey Pavao, Luis Vega, Lukas Voegtle, Maciej Bala, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Markus Kliegl, Marta Stepniewska-Dziubinska, Matthieu Le, Matvei Novikov, Mehrzad Samadi, Michael Andersch, Michael Evans, Miguel Martinez, Mike Chrzanowski, Mike Ranzinger, Mikolaj Blaz, Misha Smelyanskiy, Mohamed Fawzy, Mohammad Shoeybi, Mostofa Patwary, Nayeon Lee, Nima Tajbakhsh, Ning Xu, Oleg Rybakov, Oleksii Kuchaiev, Olivier Delalleau, Osvald Nitski, Parth Chadha, Pasha Shamis, Paulius Micikevicius, Pavlo Molchanov, Peter Dykas, Philipp Fischer, Pierre-Yves Aquilanti, Piotr Bialecki, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Rabeeh Karimi, Rahul Kandu, Ran El-Yaniv, Raviraj Joshi, Roger Waleffe, Ruoxi Zhang, Sabrina Kavanaugh, Sahil Jain, Samuel Kriman, Sangkug Lym, Sanjeev Satheesh, Saurav Muralidharan, Sean Narenthiran, Selvaraj Anandaraj, Seonmyeong Bak, Sergey Kashirsky, Seungju Han, Shantanu Acharya, Shaona Ghosh, Sharath Turuvekere Sreenivas, Sharon Clay, Shelby Thomas, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shyamala Prayaga, Siddhartha Jain, Sirshak Das, Slawek Kierat, Somshubra Majumdar, Song Han, Soumye Singhal, Sriharsha Niverty, Stefania Alborghetti, Suseella Panguluri, Swetha Bhendigeri, Syeda Nahida Akter, Szymon Migacz, Tal Shiri, Terry Kong, Timo Roman, Tomer Ronen, Trisha Saar, Tugrul Konuk, Tuomas Rintamaki, Tyler Poon, Ushnish De, Vahid Noroozi, Varun Singh, Vijay Korthikanti, Vitaly Kurin, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenliang Dai, Wonmin Byeon, Xiaowei Ren, Yao Xu, Yejin Choi, Yian Zhang, Ying Lin, Yoshi Suhara, Zhiding Yu, Zhiqi Li, Zhiyu Li, Zhongbo Zhu, Zhuolin Yang, and Zijia Chen. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models, 2025. URL https://arxiv.org/abs/2504.03624.

OpenAI. Learning to Reason with LLMs, 2025. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenThoughts. Open Thoughts. https://open-thoughts.ai, February 2025.

Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. CodeForces. https://huggingface.co/datasets/open-r1/codeforces, 2025a.

Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. IOI. https://huggingface.co/datasets/open-r1/ioi, 2025b.

Qwen. Qwen2.5: A Party of Foundation Models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

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.

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. In *COLM*, 2024.

RyokoAI. RyokoAI/ShareGPT52K. https://huggingface.co/datasets/RyokoAI/ShareGPT52K, 2023.

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.

Santhanam, Kezhi Kong, Kirthi Sivamani, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Lawrence McAf ee, Leon Derczynski, Lindsey Pavao, Luis Vega, Lukas Voegtle, Maciej Bala, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Markus Kliegl, Marta Stepniewska- Dziubinska, Matt hieu Le, Matvei Novikov, Mehrzad Samadi, Michael Andersch, Michael Evans, Miguel Martinez, Mike Chrzanowski, Mike Ranzinger, Mikolaj Blaz, Misha Smelyanskiy, Mo- hamed Fawzy, Mohammad Shoe ybi, Mostofa Patwary, Nayeon Lee, Nima Tajbakhsh, Ning Xu, Oleg Rybakov, Oleksii Kuchaiev, Olivier Delalleau, Osvald Nitski, Parth Chadha, Pasha Shamis, Paulius Micikevicius, Pavlo Molchanov, Peter Dy kas, Philipp Fischer, Pierre-Yves Aquilanti, Piotr Bialecki, Prasoon Varshney, Pritam Gundecha, Przeme k Tredak, Rabeeh Karimi, Rahul Kandu, Ran El-Yaniv, Raviraj Joshi, Roger Waleffe, Ruoxi Zhang, Sabr ina Kavanaugh, Sahil Jain, Samuel Kriman, Sangkug Lym, Sanjeev Satheesh, Saurav Muralidharan, Sean Narenthi- ran, Selvaraj Anandaraj, Seonmyeong Bak, Sergey Kashirsky, Seungju Han, Shantanu Acharya , Shaona Ghosh, Sharath Turuvekere Sreenivas, Sharon Clay, Shelby Thomas, Shrimai Prabhumoye, Shu bham Pachori, Shubham Toshniwal, Shyamala Prayaga, Siddhartha Jain, Sirshak Das, Slawek Kierat, So mshubra Majumdar, Song Han, Soumye Singhal, Sriharsha Niverty, Stefania Alborghetti, Suseella Pangu luri, Swetha Bhendigeri, Syeda Nahida Akter, Szymon Migacz, Tal Shiri, Terry Kong, Timo Roman, To mer Ronen, Trisha Saar, Tugrul Konuk, Tuomas Rintamaki, Tyler Poon, Ushnish De, Vahid Noroozi, Var un Singh, Vijay Korthikanti, Vitaly Kurin, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenliang Dai, Wonm in Byeon, Xiaowei Ren, Yao Xu, Yejin Choi, Yian Zhang, Ying Lin, Yoshi Suhara, Zhiding Yu, Zhiqi Li, Zhiyu Li, Zhongbo Zhu, Zhuolin Yang, and Zijia Chen. Nemotron-h：一系列高效且准确的混合mamba -transformer模型，2025。网址 https://arxiv.org/abs/2504.03624。

OpenAI. 学习使用大型语言模型进行推理，2025。网址 https://openai.com/index/ learning-to-reason-with-llms/。

OpenThoughts。开放思想。https://open-thoughts.ai，2025年2月。

G威廉·佩内多、安东·洛日科夫、海内克·基德利切克、卢布纳·本·阿拉尔、爱德华·比钦、阿古斯丁· 皮-克雷斯·拉哈林、昆汀·加卢德克、内森·哈比布、刘易斯·坦斯托尔，以及莱安德罗·冯·韦拉。Cod eForces。https://huggingface.co/datasets/open-r1/codeforces，2025a。

吉列尔梅·佩内多、安东·洛日科夫、海内克·基德利切克、卢布纳·本·阿拉尔、爱德华·比钦、阿古斯 丁·皮-克雷斯·拉哈林、昆汀·加卢德克、内森·哈比布、刘易斯·坦斯托尔，以及莱安德罗·冯·韦拉。IO I。https://huggingface.co/datasets/open-r1/ioi，2025b。

Q文。Qwen2.5：基础模型派对，2024年9月。网址 https://qwenlm.github. io/blog/qwen2.5/。

David Rein、Betty Li Hou、Asa Cooper Stickland、Jackson Petty、Richard Yuanzhe Pang、Julien Dira ni、Julian Michael 和 Samuel R. Bowman。Gpqa：一个研究生级别的谷歌防护问答基准，2023年。

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Juli an Michael 和 Samuel R. Bowman。GPQA：一个研究生级别的谷歌防作弊问答基准。在 *COLM*，20 24年。

RyokoAI. RyokoAI/ShareGPT52K. https://huggingface.co/datasets/RyokoAI/ShareGPT52K，2 023年。

Z邵海红，王佩怡，朱启豪，许润新，宋俊孝，毕晓，张浩伟，张明川，Y. K. Li，Y. Wu，郭大雅 。DeepSeekMath：推动开放语言模型中数学推理的极限，2024。网址 https://arxiv.org/abs/2402.03300。

Gerald Shen, Zhilin Wang, Olivier Delalleau, Jiaqi Zeng, Yi Dong, Daniel Egert, Shengyang Sun, Jimmy J. Zhang, Sahil Jain, Ali Taghibakhshi, Markel Sanz Ausin, Ashwath Aithal, and Oleksii Kuchaiev. NeMo-Aligner: Scalable toolkit for efficient model alignment. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=yK2eGE8QVW.

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.

Stack Exchange Data. Dump 2024-04-02. https://archive.org/details/stackexchange_20240402_bis, 2024.

Shengyang Sun, Yian Zhang, Alexander Bukharin, David Mosallanezhad, Jiaqi Zeng, Soumye Singhal, Gerald Shen, Adithya Renduchintala, Tugrul Konuk, Yi Dong, Zhilin Wang, Dmitry Chichkov, Olivier Delalleau, and Oleksii Kuchaiev. Reward-aware preference optimization: A unified mathematical framework for model alignment, 2025. URL https://arxiv.org/abs/2502.00203.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A Benchmark for Evaluating LLM-Based Judges. In *ICLR*, 2025.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. OpenMathInstruct-2: Accelerating AI for Math with Massive Open-Source Instruction Data. In *ICLR*, 2025.

TreeSitter. Tree sitter. https://github.com/tree-sitter/tree-sitter, 2013.

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, 2024. URL https://arxiv.org/abs/2406.01574.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. HelpSteer2: Open-source dataset for training top-performing reward models. In *ICLR*, 2025a.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Daniel Egert, Ellie Evans, Hoo-Chang Shin, Felipe Soares, Yi Dong, and Oleksii Kuchaiev. Dedicated feedback and edit models empower inference-time scaling for open-ended general-domain tasks, 2025b. URL https://arxiv.org/abs/2503.04378.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond, 2025. URL https://arxiv.org/abs/2503.10460.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Berkeley Function Calling Leaderboard. 2024.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking Benchmark and Contamination for Language Models with Rephrased Samples. 2023.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild, 2024. URL https://arxiv.org/abs/2405.01470.

Gerald Shen, Zhilin Wang, Olivier Delalleau, Jiaqi Zeng, Yi Dong, Daniel Egert, Shengyang Sun, Jimmy J. Zhang, Sahil Jain, Ali Taghibakhshi, Markel Sanz Ausin, Ashwath Aithal 和 Oleksii Kuchaiev。NeMo-Aligner：一种用于高效模型对齐的可扩展工具包。在 *First Conference on Language Modeling*，2024。网址 https://openreview.net/forum?id=yK2eGE8QVW。

穆罕默德·舒伊比、穆斯塔法·帕特瓦里、劳尔·普里、帕特里克·勒格雷斯利、贾里德·卡斯珀和布莱恩·卡坦扎罗。Megatron-LM：使用模型并行训练数十亿参数的语言模型，2020年。网址 https://arxiv.org/abs/1909.08053。

Stack Exchange 数据。数据转储 2024-04-02。https://archive.org/details/stackexchange_20240402_bis，2024年。孙胜阳、张奕、布哈林·亚历山大、莫萨拉内扎德·大卫、曾佳琪、孙穆耶·辛格尔、沈杰拉尔德、伦杜钦塔拉·阿迪西亚、科努克·图格鲁尔、董毅、王志林、奇奇·奇奇科夫、德拉勒奥·奥利维耶、库恰耶夫·奥列克谢。基于奖励的偏好优化：模型对齐的统一数学框架，2025。网址 https://arxiv.org/abs/2502.00203。

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa 和 Ion Stoica。JudgeBench：一个用于评估基于LLM的裁判的基准。在 *ICLR*，2025年。

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan 和 Igor Gitman。OpenMathInstruct-2：利用海量开源指令数据加速数学AI。在 *ICLR*，2025年。TreeSitter。树解析器。https://github.com/tree-sitter/tree-sitter，2013年。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 和 Wenhu Chen。MMLU-Pro：一个更强大、更具挑战性的多任务语言理解基准，2024年。网址 https://arxiv.org/abs/2406.01574。Z

Hilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J Zhang, Makesh Narsimhan Sreedhar 和 Oleksii Kuchaiev。HelpSteer2：用于训练顶级奖励模型的开源数据集。在 *ICLR*，2025a。

王志林、曾佳琪、奥利维耶·德拉奥、丹尼尔·埃格特、埃莉·埃文斯、申浩昌、费利佩·索阿雷斯、董毅、库恰耶夫·奥列克谢。专用反馈和编辑模型赋能推理时的规模扩展，用于开放式通用任务，2025b。网址 https://arxiv.org/abs/2503.04378。文亮、蔡云科、肖芬瑞、何新、安奇、段振宇、杜亦民、刘俊臣、唐立福、吕晓伟、邹浩生、邓永超、贾守胜、张向正。Light-r1：从零开始的课程式微调、DPO 和 RL，用于长上下文的训练，2025。网址 https://arxiv.org/abs/2503.10460。闫凡佳、毛焕志、纪成杰·查理、张天俊、Shishir G. Patil、Ion Stoica 和 Joseph E. Gonzalez。伯克利函数调用排行榜。2024。杨硕、蒋伟林、郑连民、Joseph E.. Gonzalez 和 Ion Stoica。重新思考语言模型的基准和污染问题，采用改写样本。2023。赵文婷、任翔、Jack Hessel、Claire Cardie、Choi Yejin 和邓云天。Wildchat：100万条ChatGPT在野交互日志，2024。网址 https://arxiv.org/abs/2405.01470。

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VqkAKQibpq.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

连民郑、梁生尹、志强谢、楚越孙、Jeff Huang、Cody Hao Yu、曹诗怡、Christos Kozyrakis、Ion Stoica、Joseph E. Gonzalez、Clark Barrett 和 Ying Sheng。SGLang：高效执行结构化语言模型程序。在 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*，2024。网址 https://openreview.net/forum?id=VqkAKQibpq。

J Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, 和 Le Hou。大型语言模型的指令遵循评估。*arXiv preprint arXiv:2311.07911*，2023。