Llama-Nemotron: Efficient Reasoning Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce the Llama-Nemotron series: open, heterogeneous reasoning models in three sizes—Nano (8B), Super (49B), and Ultra (253B)—that deliver strong reasoning, inference efficiency, and a permissive license. Our training pipeline combines neural architecture search, knowledge distillation, and a reasoning-focused post-training stage with supervised fine-tuning and large-scale reinforcement learning. Our large-scale RL training leverages exploration-driven curriculum and data filtering strategies to systematically challenge the model with increasingly difficult reasoning tasks, enabling it to discover and refine complex problem-solving chains beyond the capabilities of supervised learning. This approach allows the model to autonomously explore new reasoning strategies and surpass teacher performance on challenging benchmarks. Ultra achieves significantly higher GPQA accuracy and outperforms DeepSeek-R1 and other open models on key reasoning tasks. Llama-Nemotron models are also the first open-source models to support a dynamic reasoning toggle. We open-source all data, models, and code to support open research.

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 exploratory 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. Central to these advances is the use of exploration during reinforcement Learning training to discover new reasoning strategies and improve robustness.

As reasoning capabilities increasingly depend on scaling at inference time, designing models for efficient inference is essential. Inference efficiency is now a core factor for model intelligence and agentic pipelines. Equally important is giving users control over reasoning depth: not all queries benefit from detailed multi-step reasoning. The reasoning toggle enables users to actively explore different reasoning strategies and allocate resources appropriately for each task Anthropic (2025).

Here, we present the Llama-Nemotron (LN) family: open, heterogeneous reasoning models in three sizes—LN-Nano (8B), LN-Super (49B), and LN-Ultra (253B)—optimized for both inference efficiency and flexible reasoning. 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 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

Submitted to the First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025). Do not distribute.

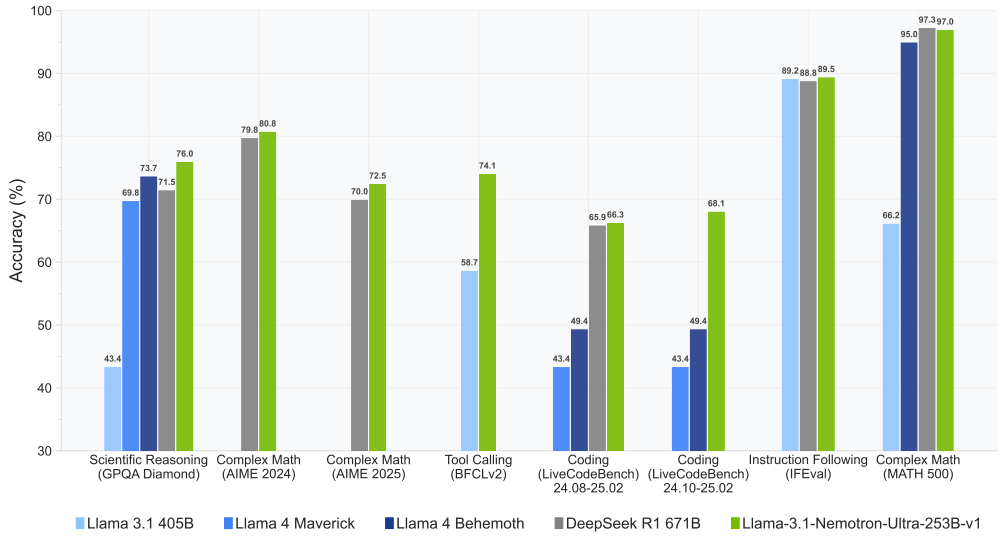


Figure 1: LN-Ultra delivers leading performance among open models across a wide range of reasoning and non-reasoning benchmarks.

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. In this report, we place particular emphasis on the post-training stages—especially our large-scale reinforcement learning runs—where exploration-driven curriculum learning and data filtering strategies are central. These approaches systematically challenge the model with progressively more difficult reasoning tasks, enabling it to autonomously discover, refine, and generalize complex reasoning behaviors that go beyond the limitations of supervised learning and are critical for achieving state-of-the-art performance on scientific reasoning benchmarks.

As part of this work, we also release 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.

2 Synthetic Data and Supervised Fine-Tuning

Modern reasoning models require not only strong reasoning capabilities but also efficient inference for practical deployment. To this end, we construct inference-optimized base models using the Puzzle neural architecture search (NAS) framework Bercovich et al. (2024), which applies techniques such as attention removal, FFN compression, and FFN fusion Bercovich et al. (2025) to compress and

accelerate Llama 3 Instruct models. This yields two model classes: LN-Super, a compressed variant of Llama 3.3-70B-Instruct that achieves 5× throughput improvement over the original model while running efficiently on a single NVIDIA H100 GPU, and LN-Ultra, derived from Llama 3.1-405B-Instruct and optimized for 8-GPU H100 nodes, delivering 1.71× latency improvements. Full details of the architecture search and optimization process are provided in Appendix A.

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. For SFT, we curate both reasoning and non-reasoning samples across math, code, science, and general domains. The key mechanism is the use of system instructions (`"detailed thinking on"` and `"detailed thinking off"`) to enable the model to toggle reasoning behavior at inference time. This setup allows the model to learn to toggle reasoning behavior at inference time based on the prompt. The data is generated using current state-of-the-art reasoning models like DeepSeek-R1, with extensive filtering and decontamination processes to ensure quality and prevent benchmark contamination. For math reasoning data, we use a pipeline described by Moshkov et al. (2025); for code reasoning, we follow the multi-stage process of Ahmad et al. (2025); and for general domain data, we use the generation pipeline from NVIDIA (2024c). The complete data curation process is provided in Appendix B. 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 summarized in Table 1 for LN-Super and Table 2 for LN-Ultra 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.

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. For detailed, model-specific SFT training procedures, please refer to Appendix C.

3 RL for Reasoning

As discussed in Section 2 and shown in Table 2, models can acquire strong capabilities through supervised fine-tuning by distilling knowledge from powerful teachers. However, distillation inherently limits the student's performance to that of the teacher, especially when the student's base model is more capable than the teacher. For example, with supervised fine-tuning, LN-Ultra can approach but not surpass the performance of DeepSeek-R1. To enable the student to exceed its teacher, we turn to large-scale reinforcement learning (RL), which empowers the model to explore new strategies and engage in self-improvement beyond imitation. This exploration-driven approach is central to advancing reasoning capabilities, as it allows the model to autonomously discover and refine complex problem-solving chains. Consistent with the findings of DeepSeek-AI et al. (2025), our preliminary experiments indicate that RL yields suboptimal results for smaller models compared to distillation. Given these observations and resource constraints, we apply reasoning RL exclusively to LN-Ultra, enabling it to surpass its teacher and set a new state-of-the-art on GPQA among open models.

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 2 shows the accuracy score on GPQA-Diamond as our training progresses. With our optimized training infrastructure (see Section D), the whole training takes about 140k H100 hours.

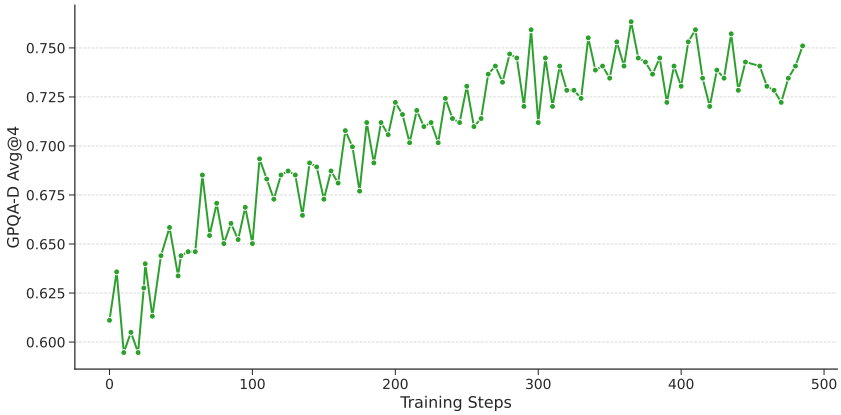


Figure 2: Accuracy on GPQA-Diamond throughout reasoning RL training for LN-Ultra.

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 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 and encouraged to explore, 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. This increases the difficulty of the training data and incentivizes exploration of less certain cases. 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 3 demonstrates the effectiveness of our curriculum strategy, which stabilizes the training process and achieves higher accuracy.

4 RL for Preference Optimization

4.1 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.

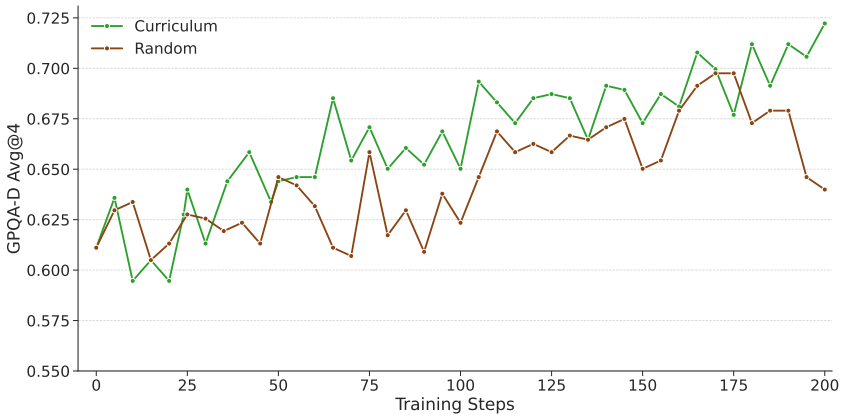


Figure 3: Curriculum training stabilizes learning and enables higher peak accuracy on GPQA-D.

4.2 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 1, LN-Super, a 49B model, achieves an 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). 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. 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.

5 Evaluations on Reasoning and Chat 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 32k context length, even though training was performed with a maximum context length of 16k for LN-Super and 24k for LN-Ultra. We observed consistent improvements when evaluating at expanded context lengths, as shorter 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

Task	LN-Super-SFT Reasoning *on*	LN-Super-SFT Reasoning *off*	LN-Super Reasoning *on*	LN-Super Reasoning *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 1: LN-Super versus comparably sized models, split by Reasoning mode.

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

LN-Nano

Although LN-Nano is the smallest model in the Llama-Nemotron series, it achieves strong performance across all reasoning benchmarks, including AIME25-I and LiveCodeBench, despite its compact size. This demonstrates the effectiveness of our SFT pipeline and curated reasoning datasets in transferring structured reasoning to smaller models. Detailed results and analysis for LN-Nano are provided in Appendix E.1.

LN-Super

Table 1 compares LN-Super to other models in its weight class where 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 based on. In reasoning-on mode, it outperforms competing models such as DeepSeek-R1-Distilled-Llama-70B, providing strong reasoning capabilities without sacrificing instruction following. Reasoning-focused SFT reduces IFEval scores, so we apply a dedicated RL stage (see Section 4.1) to restore instruction-following ability. 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.

LN-Ultra

Table 2 and Figure 1 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 2, 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*	LN-Ultra-SFT Reasoning *off*	LN-Ultra Reasoning *on*	LN-Ultra Reasoning *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 2: LN-Ultra versus the strongest open-weight models, split by reasoning mode.

6 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 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 3, 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 3: Llama-Nemotron models demonstrate strong performance on JudgeBench.

7 Conclusions

We present the Llama-Nemotron series of models which perform competitively with state-of-the-art reasoning models with efficient inference capabilities. We find that in the presence of a strong reasoning teacher, SFT 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. This stage is particularly important for enabling the model to autonomously explore new reasoning strategies and problem-solving chains, moving beyond imitation to genuine innovation.

Our results highlight that exploration, both in the form of data curation and through reinforcement learning, plays a key role in advancing model reasoning abilities. 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.

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, 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 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 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.

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.

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.

Appendix

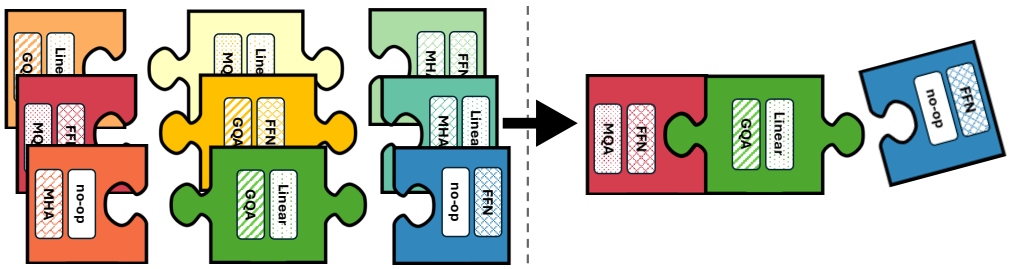
A Creating Inference-Optimized Models

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 4. 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).

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.

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



Step 1: Crafting the "puzzle pieces"
Applying block-wise local distillation to every alternative subblock replacement in parallel and scoring its quality and inference cost to build a "library" of blocks.

Step 2: Assembling the puzzle architecture
Utilizing Mixed-Integer-Programming to assemble a heterogeneous architecture that optimizes quality under constraints such as throughput, latency and memory usage.

Figure 4: Overview of the Puzzle framework.

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.

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 $5\times$ 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 $\geq 2.17\times$ throughput advantage. The model is also optimized under a constraint of approximately 300K cached tokens (batch size $\times$ 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.

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 $1.5\times$ latency reduction over Llama 3.1-405B-Instruct. After applying FFN Fusion, the final model achieves a $1.71\times$ 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 5 illustrates the trade-off between GPQA-Diamond accuracy (%) and processing throughput (tokens/s) for two settings. Notably, LN-Ultra consistently outperforms 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.

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).

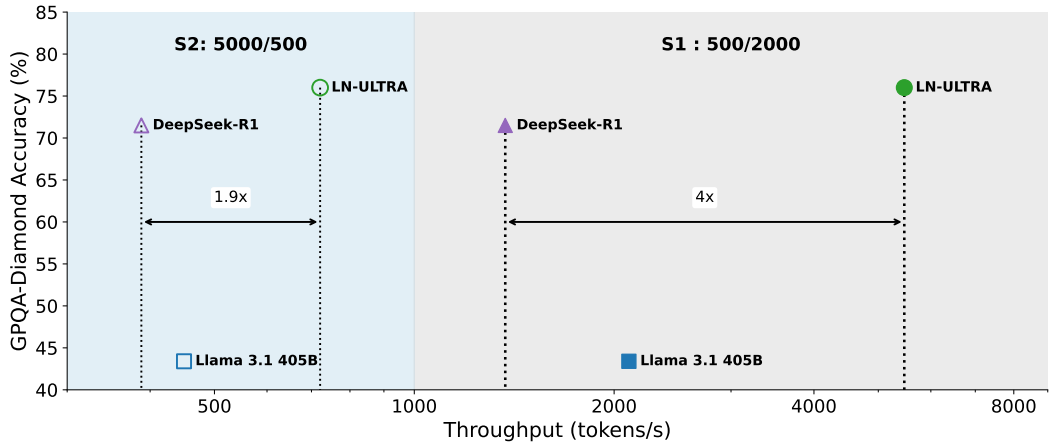


Figure 5: GPQA-Diamond Accuracy vs. Throughput. We measure on two settings, S1: 500/2000 (ISL/OSL); S2: 5000/500 (ISL/OSL). Both with 250 concurrent users. Models are served with FP8. Note that we use 8×H100 for LN-Ultra and Llama 3.1 405B, but 8×H200 for Deepseek-R1 because of its size.

- 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 **??**).

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

B 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.

B.1 Reasoning on

B.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.

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.

B.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.

B.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.

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.

B.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.

B.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 B.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 B.3.

B.3 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.

The detailed composition of our synthetic dataset is shown in Table 5.

B.4 Synthetic Data Composition

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 5: Synthetic data by domain with reasoning splits.

C Model-Specific SFT Training Details

LN-Nano differently from other models below, undergoes a three-stage SFT pipeline using a global batch size of 256 using sequence packing with effective sequence length of 32k tokens. In the first stage, the model is fine-tuned exclusively on reasoning data from code, math, and science domains (Section B.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 B.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 suggested 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 training 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.

D Infrastructure

D.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. The generation model parallelism was tensor parallel=8, and data parallel=72. The details of how this parallelization strategy is chosen is explained in D.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.

D.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 D.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 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 D.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.

D.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.

E Additional Results and Analysis

E.1 LN-Nano Detailed Results

Table 6 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 LN-Nano, carefully balancing data-distribution across math, coding, and stem areas has been important to achieve near state-of-the-art accuracies at the SFT stage. For example, our early experiments showed worse accuracies especially in chemistry related questions, 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 6.

Task	**LN-Nano-SFT** Reasoning		**LN-Nano** Reasoning		**DeepSeek-R1 Distilled-Llama-8B**	**Llama-3.1 8B-Instruct**	**DeepSeek-R1 Distilled Qwen-7B**
	on	***off***	***on***	***off***			
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 6: LN-Nano and LN-Nano-SFT versus comparably sized models, split by Reasoning mode.

E.2 Artificial Analysis Comparison

According to Artificial Analysis (shown in Figure 6), 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.

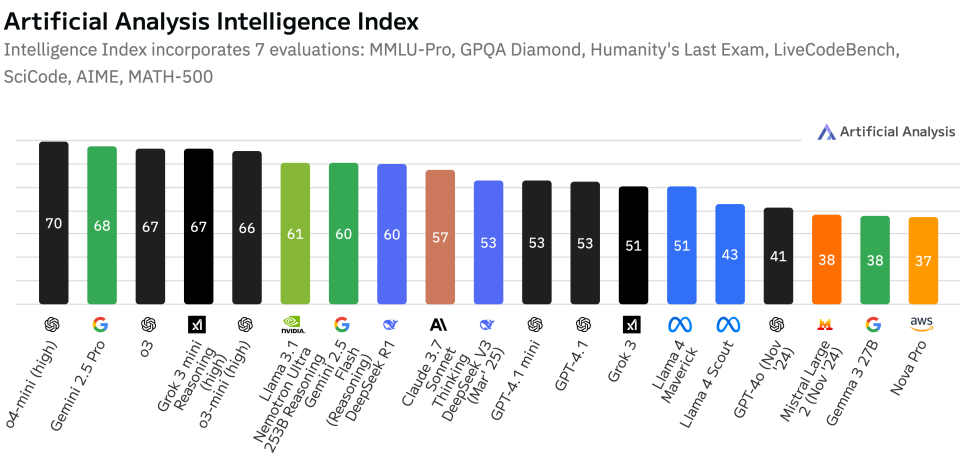


Figure 6: As of April 2025, our flagship model LN-Ultra is the most "intelligent" open model according to Artificial Analysis.