Nemotron 3 Nano: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning

NVIDIA

Abstract. We present Nemotron 3 Nano 30B-A3B, a Mixture-of-Experts hybrid Mamba-Transformer language model. Nemotron 3 Nano was pretrained on 25 trillion text tokens, including more than 3 trillion new unique tokens over Nemotron 2, followed by supervised fine tuning and large-scale RL on diverse environments. Nemotron 3 Nano achieves better accuracy than our previous generation Nemotron 2 Nano while activating less than half of the parameters per forward pass. It achieves up to 3.3× higher inference throughput than similarly-sized open models like GPT-OSS-20B and Qwen3-30B-A3B-Thinking-2507, while also being more accurate on popular benchmarks. Nemotron 3 Nano demonstrates enhanced agentic, reasoning, and chat abilities and supports context lengths up to 1M tokens. We release both our pretrained Nemotron 3 Nano 30B-A3B Base and post-trained Nemotron 3 Nano 30B-A3B checkpoints on Hugging Face.

1. Introduction

We present NVIDIA Nemotron 3 Nano, a Mixture-of-Experts (MoE) hybrid Mamba-Transformer model (Lieber et al., 2024) with agentic, reasoning, and chat capabilities. Like previous generations (NVIDIA, 2025e,d), Nemotron 3 Nano uses a combination of Mamba-2 (Dao & Gu, 2024) and Grouped-Query-Attention (GQA) (Ainslie et al., 2023). In addition, Nemotron 3 Nano uses Mixture-of-Experts (Shazeer et al., 2017) layers to scale model parameters sparsely and achieve significant improvements on the inference-throughput-to-accuracy frontier. We use a granular MoE architecture (Dai et al., 2024) with a learnt MLP router that activates 6 out of 128 experts (§2.1). Nemotron 3 Nano totals 31.6B parameters out of which only 3.2B are activated per forward pass (3.6B including embeddings). Nemotron 3 Nano achieves better or on-par accuracy compared to GPT-OSS-20B (OpenAI, 2025) and Qwen3-30B-A3B-Thinking-2507 (Yang et al., 2025a) as shown in Figure 1. Further, on the 8K input / 16K output token scenario, Nemotron 3 Nano provides 2.2× and 3.3× faster inference throughput compared to GPT-OSS-20B and Qwen3-30B-A3B-Thinking-2507 respectively. Nemotron 3 Nano also supports context lengths up to 1M tokens, outperforming both GPT-OSS-20B and Qwen3-30B-A3B-Instruct-2507 on RULER across different context lengths. Along with the model weights, we provide the recipe, code, and most of the data we used to train the model.

We pretrained our base model, Nemotron 3 Nano 30B-A3B Base, using the Warmup-Stable-Decay (Hu et al., 2024) learning rate schedule on 25 trillion tokens of text data spanning 15 categories (§2.2). We divided pre-training into 2 phases with 23.5 trillion tokens of diverse data in the first phase, followed by 1.5 trillion tokens of high-quality data in the second phase (§2.3). Our base model achieves better accuracy than equivalent-sized Qwen3-30B-A3B-Base on most academic benchmarks across Code, Math, Long Context, General Knowledge, and Commonsense Understanding categories. We do not compare the accuracy of our base model to GPT-OSS-20B because no base model was released with it. Our model also achieves significantly better inference throughput than Qwen3-30B-A3B (3.3×) and GPT-OSS-20B (2.2×) on generation heavy 8K input / 16k output scenario when tested on a single H200 GPU. We measured throughput using the best configuration available for H200 GPUs

© 2025 NVIDIA. All rights reserved.

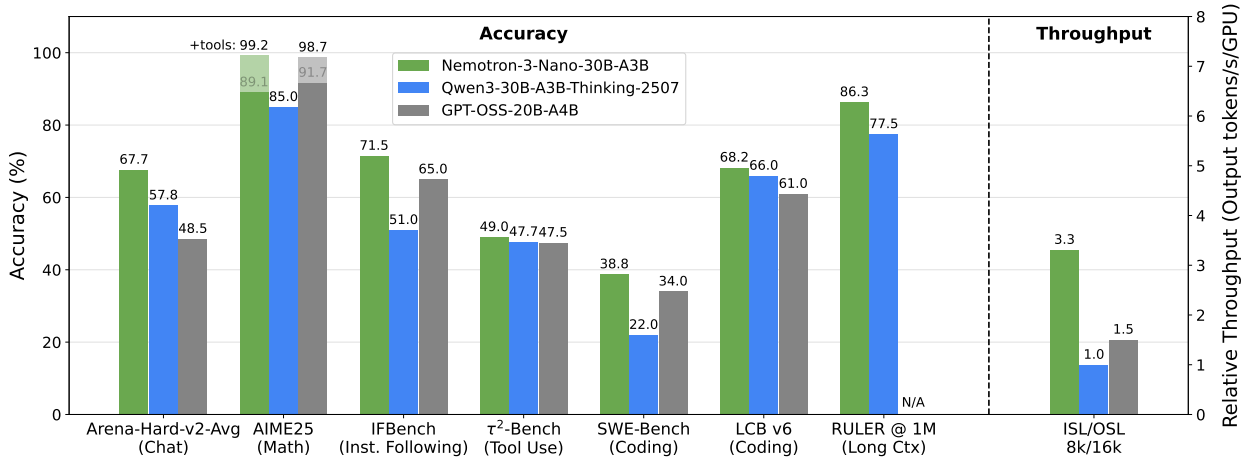


Figure 1 | Accuracy and throughput comparisons of Nemotron 3 Nano with Qwen3-30B-A3B-Thinking-2507 and GPT-OSS-20B. Nemotron 3 Nano achieves on-par or better accuracies across multiple benchmarks. RULER scores for 1M context length are available only for Nemotron 3 Nano and Qwen3 since GPT-OSS-20B has a context length of 128K tokens. Further, on 8K input / 16K output setting, Nemotron 3 Nano provides inference throughput that is 3.3× higher than Qwen3-30B-A3B-Thinking-2507 and 2.2× higher than GPT-OSS-20B. We measured throughput on a single H200 GPU with vLLM and TRT-LLM and used the best out of the two for each model. We used the OpenHands harness to evaluate SWE-Bench.

with both vLLM and TRT-LLM and used the better of the two for each model. We used `FP8` for both weights and activations for throughput measurement of Nemotron 3 Nano and Qwen3. We used `mxfp4` for weights and `bfloat16` for activations for GPT-OSS-20B.

We post-trained Nemotron 3 Nano using three approaches: supervised fine tuning (SFT) (§3.1), multi-environment reinforcement learning from verifiable rewards (RLVR) (§3.2), and reinforcement learning from human feedback (RLHF) (§3.3). During SFT, we trained Nemotron 3 Nano on a diverse set of chat, agentic, and reasoning traces to imbue the model with reasoning budget control, reasoning on/off control, and tool-integrated reasoning capabilities. During RLVR, we trained on all environments simultaneously, resulting in a smooth and uniform improvement in model capabilities. During RLHF, we utilized a large and accurate generative reward model (GenRM) to enhance the performance of Nemotron 3 Nano on key chat benchmarks.

We also quantized Nemotron 3 Nano from `bfloat16` to `FP8` using post training quantization (PTQ). This helps achieve higher inference throughput with minimal loss in accuracy (§4.3).

Along with this report, we are releasing the model recipes[1] and publishing the following:

Checkpoints

- `Nemotron 3 Nano 30B-A3B FP8` 🤗 : the final post-trained and FP8 quantized model
- `Nemotron 3 Nano 30B-A3B BF16` 🤗 : the post-trained model
- `Nemotron 3 Nano 30B-A3B Base BF16` 🤗 : the pre-trained base model
- `Qwen-3-Nemotron-235B-A22B-GenRM` 🤗 : the GenRM used for RLHF

Data

[1] https://github.com/NVIDIA-NeMo/Nemotron

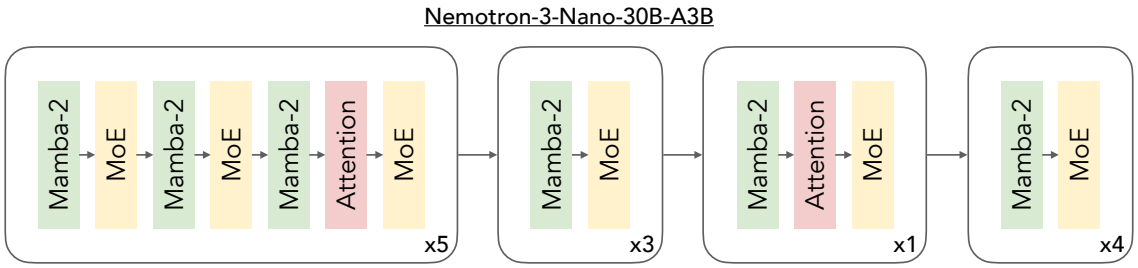


Figure 2 | Nemotron 3 Nano layer pattern. We use a hybrid Mamba-Transformer architecture similar to the previous generation of Nemotron models. In addition, we scale the model sparsely by using MoE layers instead of standard FFN layers.

- `Nemotron-CC-v2.1` 🤗 : 2.5 trillion new English tokens from Common Crawl, including curated data from 3 recent snapshots, synthetic rephrasing, and translation to English from other languages.
- `Nemotron-CC-Code-v1` 🤗 : A pretraining dataset consisting of 428 billion high-quality code tokens obtained from processing Common Crawl Code pages using the Lynx + LLM pipeline from `Nemotron-CC-Math-v1` . Preserves equations and code, standardizes math equations to LaTeX, and removes noise.
- `Nemotron-Pretraining-Code-v2` 🤗 : Refresh of curated GitHub code references with multi-stage filtering, deduplication, and quality filters. Large-scale synthetic code data.
- `Nemotron-Pretraining-Specialized-v1` 🤗 : Collection of synthetic datasets for specialized areas like STEM reasoning and scientific coding.
- `Nemotron-SFT-Data` 🤗 : Collection of new Nemotron 3 Nano SFT datasets.
- `Nemotron-RL-Data` 🤗 : Collection of new Nemotron 3 Nano RL datasets.

We divide the remainder of the report into 3 sections: Pre-training (§2), Post-Training (§3), and Quantization (§4).

2. Pretraining

In this section, we highlight the key features of Nemotron 3 Nano 30B-A3B Base, including its architecture, hyperparameters, and the data used for pretraining. We also show that Nemotron 3 Nano 30B-A3B Base achieves better accuracy than other public state-of-the-art models across a suite of benchmarks.

2.1. Model Architecture

Nemotron 3 Nano 30B-A3B Base builds upon the hybrid Mamba-Transformer architecture of our older Nemotron-H (NVIDIA, 2025e) and Nemotron 2 Nano (NVIDIA, 2025d) models by replacing the standard FFN layers with sparse Mixture-of-Experts (MoE) (Shazeer et al., 2017) layers. The MoE layers help us achieve better accuracy at a fraction of the active parameter count. Nemotron 3 Nano 30B-A3B Base contains 31.6B total parameters out of which 3.2B are active (3.6B including embeddings) per forward pass. To achieve the best accuracy, we use a granular MoE architecture along with shared experts (Dai et al., 2024). For the MoE layers, we use squared ReLU activation and a standard learnt MLP router with sigmoid gating. We do not use any positional embeddings, dropout, or bias on linear layers. We use RMSNorm for normalization and un-tie embedding and projection weights.

Table 1 and Figure 2 show the key architectural details of Nemotron 3 Nano.

Model	Nemotron 3 Nano 30B-A3B Base
Num Layers	52
Model Dimension	2688
Q-heads	32
KV-heads	2
Head Dimension	128
Mamba State Dimension	128
Mamba Groups	8
Mamba Heads	64
Mamba Head Dimension	64
Expert Dimension	1856
Total Routable Experts	128
Number of Activated Experts	6
Number of Shared Experts	2

Table 1 | Nemotron 3 Nano Architecture

2.2. Pretraining Data

In this sub-section, we describe new datasets that we added to our pretraining corpus since Nemotron Nano 2. We are releasing the vast majority of the new data on HuggingFace, divided into four main datasets. We describe each of these in more detail below.

2.2.1. Nemotron-CC-Code-v1

We first filtered out the code pages in Common Crawl based on a fast pattern matching code classifier for webpages. We then constructed our high-quality code pretraining corpus by applying a modified version of the Nemotron-CC-Math pipeline (Mahabadi et al., 2025) to Common Crawl pages containing code.

Starting from raw HTML, we rendered each document using Lynx, which reliably preserved code layout, indentation, and inline technical elements. The resulting text was processed by an LLM-based cleaning stage using the Phi-4 model, which removed boilerplate while strictly retaining code snippets, configuration blocks, API references, and mathematical expressions. To ensure that only programming-relevant documents are included, we applied a lightweight code-quality relevance classifier, filtering out non-technical pages and retaining documents with substantial or complete code content. This pipeline produced a 427.92B-token corpus in which equations are standardized to LaTeX, code blocks are preserved with structural fidelity, and noise is minimized. Compared to previous extraction approaches that often corrupt or truncate code examples, our method reliably recovered complete code snippets and technical context at scale.

2.2.2. Nemotron-Pretraining-Code-v2

We sourced additional code from GitHub for repositories we identified as missing from our existing corpus in addition to collecting recent data with a cut-off date of April 15, 2025. We used the same pipeline as described in NVIDIA (2025d) to curate the data and we remove exact and near-duplicate files already present in our existing corpus.

In addition to our raw source-code corpus, we synthetically generated additional mixed natural-language and source code documents using the Qwen3 32B LLM. Similar to our approach described in NVIDIA (2025e), we prompted the model to generate question and answer pairs using our new source-code data as seeds. Additionally, we prompted the model to generate student-teacher (Python only) and code-review (Python/C++) style dialogue grounded with a combination of code snippets and full source files.

Following the code-rewriting work presented in Fujii et al. (2025), we also found that using LLMs to rephrase source code improved downstream code-generation accuracies. Using Qwen3 32B, we rephrased all of our raw Python source code using a combination of the Style-Guided Code Rewriting (SGCR) and Self-Contained Optimization Rewriting (SCOR) prompts (Fujii et al., 2025), as well as our own prompt with similar intent. To ensure high-quality LLM rephrasing, as a post-processing step, we checked for syntax errors and assessed code-quality improvements using the Pylint Python linter for each of the rewritten files.

While LLM-based source-code rewriting can be observed as a transformation of the original source-code to an improved version, we extended this concept and applied it to source-code files from one language to another (i.e., code transpilation). Using Qwen3 32B we found that C++ tokens produced from Python using this transpilation procedure improved downstream C++ code-generation accuracy and thus served as a useful augmentation to our C++ subset. We applied this Python to C++ transpilation procedure to all Python source files in our source-code corpus.

2.2.3. Nemotron-CC-v2.1

For general English web crawl data, we added three more recent Common Crawl snapshots on top of https://huggingface.co/datasets/nvidia/Nemotron-CC-v2 (CC-MAIN-2025-18, CC-MAIN-2025-21, CC-MAIN-2025-26), prepared with the same Nemotron-CC recipe (Su et al., 2025). For all of the synthetic rephrasing, we used Qwen3-30B-A3B (Yang et al., 2025a). Just as for Nemotron Nano 2, we trained only on the Medium-Quality, Medium-High-Quality, and High-Quality buckets.

Previously, we rephrased only the High-Quality subset of Common Crawl data. To further expand our corpus of unique high-quality tokens, we applied five prompts (Su et al., 2025) to the Medium-High-Quality data from 110 Common Crawl snapshots (CC-MAIN-2013-20 - CC-MAIN-2025-26), resulting in 2.1T new tokens.

Finally, we employed a new strategy to source high-quality English tokens by translating to English from other languages using Qwen3-30B-A3B. We first translated documents from the latest three Common Crawl snapshots available at that time (CC-MAIN-2024-51, CC-MAIN-2025-08, and CC-MAIN-2025-18) in 9 languages (Chinese, French, German, Italian, Japanese, Polish, Portuguese, Russian, Spanish) to English. After that, we applied the Nemotron-CC ensemble of quality classifiers to retain only High-Quality and Medium-High-Quality documents from this translated subset. Additionally, we applied four of the five Nemotron-CC rephrasing prompts to the high-quality data to generate more unique tokens. After training of Nemotron 3 Nano 30B-A3B Base was already underway, we found that some uninformative translated documents (e.g., daily conversations, ads) were receiving high scores from the Nemotron-CC quality classifiers. To address this, for the released version of this dataset, we performed one additional pass of LLM-based quality filtering that removed approximately 10.6 % of tokens, which slightly improved accuracies across benchmarks in an internal ablation.

Overall, we curated or generated over 2.5T new tokens from Common Crawl data.

2.2.4. Nemotron-Pretraining-Specialized-v1

This dataset comprises various synthetic datasets that are specialized for specific topics like STEM Reasoning or scientific coding. We describe the subsets in more detail below.

Synthetic Wikipedia Data We revised English Wikipedia articles using Qwen3-30B-A3B-Instruct-2507 to improve clarity and formatting. We discarded disambiguation and redirect pages and removed References, See also, Notes, and External Links sections. We also instructed the model to remove any irrelevant content such as uncleaned HTML elements.

Synthetic Math Textbook Data We generated well-structured educational textbook-style sections from Nemotron-CC-Math (Mahabadi et al., 2025). We evaluated the mathematical content in each document and classify it into an educational level (e.g., grade school, middle school, high school) based on multiple factors such as involved mathematical concepts and complexity. We kept documents containing mathematical content at the undergraduate level and above and developed each into a textbook-style section with diverse educational features such as definitions and illustrative examples.

Synthetic Scientific Coding Data Using STEM-related documents retrieved from Nemotron-CC as the seed data, we synthesized two types of documents: (1) Code-embedded article: A comprehensive, in-depth, and well-formatted article that explores and implements a non-trivial, graduate- or research-level scientific or mathematical algorithm in Python; (2) Computational coding problem: An advanced, computational, graduate- or research-level coding problem with Python solution. The main problem is decomposed into 5 to 15 logically ordered non-trivial substeps, each solved by an individual function. We extract the main problem, dependencies, substep descriptions, and each function's signature, docstring, body, and return statement and exclude examples where any of these components are missing.

Synthetic Cross-Domain Code Data To generate more diverse and complex code data, we develop a novel approach we call *InfiniByte* that cross-breeds multiple datasets together. When applied to code, InfiniByte creates entirely new programming problems by bringing together concepts from different fields to pose never before seen questions. In doing so, InfiniByte fills the problem space between disparate domains, generates questions at the boundary of model capabilities, and mimics how science is often advanced at the intersection of two or more fields.

Starting with a curated list of competitive coding problems from our groundbreaking OpenCodeReasoning dataset (Ahmad et al. (2025b)), we systematically inject concepts from datasets across mathematics (OpenMathReasoning, Moshkov et al. (2025)), physics (Physics Big, Zaharov et al. (2024)), chemistry (IChO, Nguyen (2025)), and other sciences. We generate multiple problem candidates per (problem, concept) combination, select the best problem candidate, based on LLM-as-critic rubric that tests for clarity, difficulty, and adherence to the employed cross-breeding strategy. We then generate solutions to each new coding problem using a reasoning model such as `Qwen3-235B-A22B-Thinking-2507` (Yang et al., 2025a). We cross-breed with two different strategies in mind:

1. Obfuscate without really changing the original problem (this is common in competitive coding problems and other competitions).
2. Complicate by actually making the new problem much more complex: the resulting problem is more challenging as it requires reasoning across multiple concepts to solve it.

The InfiniByte data generation pipeline was implemented in NeMo Data Designer (The NeMo Data Designer Team (2025)), NVIDIA's state-of-the-art synthetic data generation framework. This

allowed our complex pipeline to benefit from the compound AI approach of the framework in order to enforce proper concept grounding via Jinja templating, guarantee structured outputs required at all stages, incorporate feedback loops, as well as perform data validation and automated retries.

Synthetic STEM Reasoning To reinforce complex reasoning capabilities within STEM domains, we built the Reasoning Question-Answer (RQA) dataset. Our goal in the creation of RQA was two-fold:

i) Demonstrate advanced scientific reasoning and instruction following that can be further reinforced in post-training, as shown in Akter et al. (2025).
ii) Reinforce correlations between advanced topics that are otherwise rarely observed in web-scale data.

The dataset was generated in four steps. First, we targeted diverse and advanced scientific texts as seed data. Starting from the STEM subset of the Essential-Web web-scale dataset (Hojel et al., 2025), we filtered the dataset using the Essential-Web taxonomy to documents that met the following criteria:

- Undergraduate or graduate education level.
- No extraction artifacts, no missing content.
- Advanced reasoning depth.
- High or exceptional technical correctness.
- Leverages one of the Bloom cognitive processes: *Analyze*, *Evaluate* or *Create*.
- Leverages one of the Bloom knowledge domains: *Conceptual*, *Procedural* or *Metacognitive*.
- In the English language and over 1000 characters.

This filtering resulted in approximately 14 million documents. Next, we used hierarchically stratified sampling on document topics to trade-off between seed document volume and diversity. Leveraging the Free Decimal Correspondence (FDC) numerical topic code from the Essential-Web taxonomy, documents were ordered in hierarchical round-robin fashion across multiple orders of magnitude in the FDC code, from high-level topic domains (e.g. Physics, Chemistry, Math, Computer Science) to lower-level subdomains (e.g. Thermodynamics, Quantum Mechanics). Using this approach, we could apply any cutoff N to the seed documents to ensure maximum diversity for a given volume of documents; while we generated RQA samples for the first 9 million samples, we ultimately chose to use the first 4.5 million for training. To limit the length of each seed document, we post-processed documents over 4096 characters in length to extract a random contiguous text chunk consisting of $<$4096 characters.

Each seed document was presented as context to `Qwen3-235B-A22B-Thinking-2507`, which was prompted to use the STEM content as inspiration for a difficult (yet answerable) graduate-level scientific reasoning question. The model was instructed to ensure that the question did not require access to the original seed passage to answer. Examples were discarded if they failed to produce a question within 8192 reasoning tokens.

Finally, this question was presented to `Qwen3-235B-A22B-Thinking-2507` to answer in a second generation step, without including the seed passage as context. The resulting reasoning trace and answer were filtered to remove model-specific idiosyncrasies, limited to 8192 characters, and concatenated with the question to produce a single RQA example. The two-step generation process was designed to maximally engage the teacher model's reasoning capabilities, both in generating a difficult question from the seed document and in answering its own question. The resulting

pretraining dataset consists of 4.3 million RQA demonstrations for a total of approximately 31.7 billion unique tokens.

To make further use of the stratified STEM seed documents, we also produced a diverse QA (DQA) version of the dataset, using the first 9 million seed documents in stratification order for a total of approximately 8 billion tokens. The STEM DQA dataset was built by using the DQA prompt & generation procedure as demonstrated in Nemotron-CC (Su et al., 2025), which concatenates a contiguous text chunk from the source document with short-form question-answer pairs. We utilized `Qwen3-30B-A3B` to generate these QA pairs.

Both RQA and DQA data generation pipelines were implemented in NeMo Data Designer (The NeMo Data Designer Team (2025)).

SFT-style data. We included new and refreshed SFT datasets in pretraining for code, math, and STEM, just as for Nemotron Nano 2. Detailed synthesis methods and pipelines can be found in prior works (Toshniwal et al., 2024; Moshkov et al., 2025; NVIDIA, 2025a; Ahmad et al., 2025b,a; Majumdar et al., 2024). We also incorporated a set of additional math and code SFT samples from AceReason-Nemotron-1.1 (Liu et al., 2025a). This collection encompasses a wide range of prompt sources, including NuminaMath (Li et al., 2024b), OrcaMathWordProblems (Mitra et al., 2024), MathInstruct (Yue et al., 2023), and MetaMathQA (Yu et al., 2023) for math tasks, as well as TACO (Li et al., 2023), APPs (Hendrycks et al., 2021), OpenCoder-Stage2 (Huang et al., 2024), and OpenCodeReasoning (Ahmad et al., 2025b) for coding tasks. The responses for these prompts are generated by DeepSeek-R1 (DeepSeek-AI, 2025a).

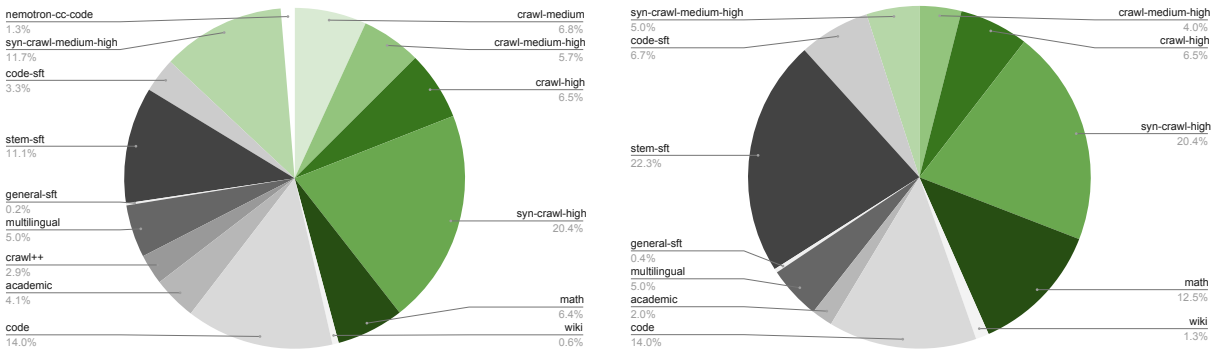
2.3. Data Mixture and Ordering

Our pretraining corpus spans fifteen data categories. The largest component is web crawl data, which we subdivide into five quality-based groups following the Nemotron-CC taxonomy (Su et al., 2025): crawl-medium, crawl-medium-high, syn-crawl-medium-high, crawl-high, and syn-crawl-high, representing medium, medium-high, high, crawl data. Beyond web crawl, the mixture also includes math, Wikipedia, code, nemotron-cc-code, academic text, Crawl++, multilingual data, and synthetic SFT-style datasets, the latter further grouped into general-sft, stem-sft, and code-sft categories. Crawl++ comprises the OpenWebText, BigScience and Reddit datasets. Our multilingual data has nineteen languages: Arabic, Chinese, Czech, Danish, Dutch, Finnish, French, German, Hebrew, Hindi, Italian, Japanese, Korean, Portuguese, Polish, Russian, Spanish, Swedish, and Thai. We design our data mixtures to balance coverage and quality by assigning comparable weight to sources of similar estimated quality. Higher-quality datasets are prioritized accordingly, receiving greater weight in the blend. Additional details on our dataset quality assessment and mixture construction methodology can be found in Feng et al. (2024) and NVIDIA (2025e).

We used a curriculum based on two phases to pre-train Nemotron 3 Nano 30B-A3B Base. In the first phase, we used a data mixture that promotes diversity in data; in the second phase, we primarily use high-quality datasets (e.g., Wikipedia). We switched to the second phase at the 94% point of training. The data mixtures used in each phase are shown in Figure 3.

2.4. Hyperparameters

We pretrained Nemotron 3 Nano 30B-A3B Base using the Warmup-Stable-Decay learning rate (LR) schedule for a total of 25 trillion tokens. We warmed up the LR over 8.4 billion tokens to a maximum of 10^{-3}. We maintained the maximum LR for 80% of training (20 trillion tokens) and then finally decayed to a minimum of 10^{-5} during the last 20% of training (5 trillion tokens). We used the AdamW (Loshchilov & Hutter, 2017) optimizer with weight decay of 0.1, $\beta_1 = 0.9$, and $\beta_2 = 0.95$.



(a) Data mixture of Phase 1.

(b) Data mixture of Phase 2.

Figure 3 | Data mixtures for each phase of pre-training.

We pretrained the model with a sequence length of 8192 and a batch size of 3072, resulting in roughly 25 million tokens per batch. For the MoE layers, we used DeepSeek's aux-loss-free load balancing strategy (Wang et al., 2024; DeepSeek-AI, 2025b) with an update rate of 10^{-3} in conjunction with the standard load balancing loss (Lepikhin et al., 2020). We used a load balancing loss coefficient of 10^{-4}.

2.5. Long-Context Extension

Similar to Nemotron 2 Nano, we added a long-context phase (LC-Phase) at the end of pretraining. In the LC-Phase, we performed continuous pretraining (CPT) to equip the base model with long-context ability. We used a constant learning rate of 10^{-5} and global batch size of 48. We used 8-way context parallelism, 8-way tensor parallelism, 8-way expert parallelism, and 4-way pipeline parallelism to train on H100 GPUs. We reused the long-context document QA dataset from Nemotron Nano 2, but scaled it to make it 3× larger. We also added a small amount of synthetic retrieval-focused data to the CPT data blend, with a maximum sequence length of 256k tokens, to help improve subset of RULER style tasks. We allocated the document QA and synthetic retrieval-focused data to 20% and 1% in the Phase LC data blend, with the remaining 79% being downscaled Phase 2 data. We initially tried performing CPT on data batches with only sequence lengths of 524,288 (512k) tokens, but found that short-context benchmark scores were impacted to a small extent. Consequently, we used a mixture of 512k and 4k sequences, which resulted in improved short-context benchmark scores, especially MMLU-Pro and Code, while also improving long-context benchmark scores. LC-Phase used a total of 121 billion tokens.

2.6. Base Model Evaluations

Table 2 presents a comprehensive accuracy comparison across general knowledge, code, math, commonsense understanding, reading comprehension, multilingual, and long context benchmarks. Evaluation settings adhered to standard community protocols to ensure fair comparison. All evaluation results were collected via Nemo Evaluator SDK[2] and LM Evaluation Harness[3]. For reproducibility purposes, more details on the evaluation settings can be found in the Nemo Evaluator SDK configs folder[4], and the open source container on LM Evaluation Harness packaged via NVIDIA's

[2] https://github.com/NVIDIA-NeMo/Evaluator
[3] https://github.com/EleutherAI/lm-evaluation-harness
[4] https://github.com/NVIDIA-NeMo/Evaluator

Task	Qwen3-30B A3B-Base	N-3-Nano 30B-A3B Base
General Knowledge		
MMLU (5-shot, acc)	**81.07**	78.56
MMLU-Pro (5-shot, CoT EM)	61.71	**65.05**
AGIEval-En (3/5-shot, CoT acc)	63.12	**68.32**
Code		
HumanEval (0-shot)	70.73	**78.05**
MBPP-Sanitized (3-shot)	73.15	**75.49**
Math		
GSM8K (8-shot, acc)	89.01	**92.34**
MATH (4-shot, acc)	61.14	**82.88**
MATH-500 (4-shot, avg@32)	55.08	**78.63**
Commonsense Understanding		
ARC-Challenge (25-shot, acc_norm)	**94.45**	91.89
HellaSwag (10-shot, acc_norm)	83.14	**85.56**
OpenBookQA (0-shot, acc_norm)	44.80	**46.20**
PIQA (0-shot, acc_norm)	81.01	**84.33**
WinoGrande (5-shot, acc)	78.22	**79.64**
Reading Comprehension		
RACE (0-shot, acc)	**90.05**	88.04
Multilingual		
MMLU Global Lite (5-shot, avg acc)	**76.84**	74.47
MGSM (8-shot, avg acc)	82.53	**83.00**
Long Context		
RULER (64K, 0-shot, acc)	63.55	**87.50**
RULER (128K, 0-shot, acc)	60.69	**82.92**
RULER (256K, 0-shot, acc)	-	**75.44**

Table 2 | Comparison of **Qwen3-30B-A3B-Base** and **Nemotron 3 Nano 30B-A3B Base**. Best results are marked in bold.

Nemo Evaluator SDK used for evaluations can be found here[5].

For the MATH-500 task, we employed a sampling strategy to report the `avg@32` score (`pass@1` estimated from 32 samples). For the rest of the tasks, we report accuracy (`acc`) or normalized accuracy (`acc_norm`) obtained via greedy decoding (`temperature = 0`). For code evaluations, HumanEval and MBPP, we apply the same sanitization method as in Evalplus[6]. Few-shot settings varied by benchmark, ranging from 0-shot for HumanEval to 25-shot for ARC-Challenge. Multilingual capabilities were evaluated on MMLU Global Lite (averaging across German, Spanish, French, Italian, Japanese, Korean, Portuguese, and Chinese) and MGSM (averaging across German, Spanish, French, Japanese, Russian, and Chinese).

To gain deeper insights into the model's capabilities, we further evaluate the model on two variants of MMLU-redux (See Appendix B).

3. Post-Training

In comparison to Nemotron Nano 2, we significantly scale up the compute in post-training for Nemotron 3 Nano. Noticeably Nemotron 3 Nano is our first effort to scale up reinforcement learning (RL) in the post-training stage. This RL scale up is empowered by multi-environment reinforcement learning (discussed in Sections 3.1 to 3.3), where we train on all environments simultaneously for the first time. We adopted Nemo-Gym, a RL training environment orchestration framework with a large collection of RL environments; this is integrated with Nemo-RL as the RL training framework as discussed in Section 3.2.4 (NVIDIA, 2025b,c). We open source Nemo-Gym and Nemo-RL to enable the broader community to facilitate large-scale RL training, as well as collaborative and distributed RL environment building.

In the rest of this section, we discuss the post-training methodology for Nemotron 3 Nano, which includes supervised finetuning (SFT) in §3.1, multi-environment reinforcement learning in §3.2, and reinforcement learning from human feedback (RLHF) in §3.3. The final evaluation results can be found in §3.4. Our post-training methodology results in best-in-class performance in a variety of reasoning and agentic tasks, along with token efficiency, reasoning on/off control, reasoning budget control, and tool-integrated reasoning capabilities.

3.1. Supervised Fine Tuning

Since the release of Nemotron 2 Nano, we have significantly improved our SFT strategy. We increased dataset quality and diversity, adding a wide variety of new data with an emphasis on multi-step and multi-turn agentic tasks. Different from the SFT data in the pre-training stage, the SFT stage data is more focused on agentic tasks and has the chat-template applied. We release the majority of our training data and open source our SFT codebase.

3.1.1. Chat Template

We allow using Nemotron 3 Nano in reasoning or non-reasoning mode through the chat template.

In reasoning mode, we alter the reasoning flow for the following conversation scenarios:

- *Multi-Step*: In a series of assistant model calls, the existing reasoning tokens are preserved to allow the model to re-use existing reasoning for subsequent step.

[5] https://catalog.ngc.nvidia.com/orgs/nvidia/teams/eval-factory/containers/lm-evaluation-harness
[6] https://github.com/evalplus/evalplus

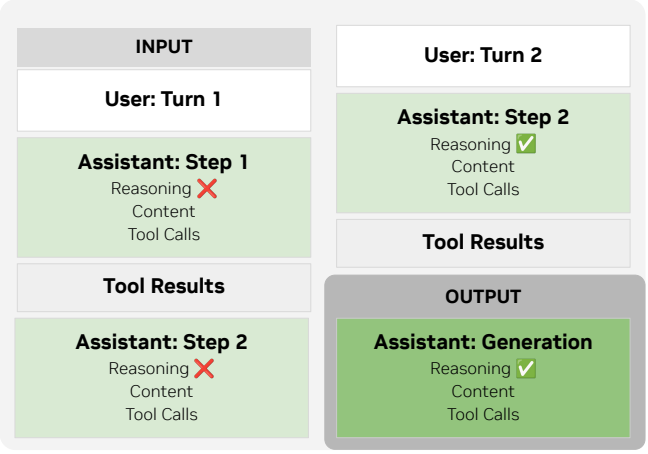


Figure 4 | Example prompt materialization using the Nemotron 3 Nano chat template for a 2-turn conversation. For a given generation, only reasoning content from the current turn is materialized into the prompt.

- *Multi-Turn*: When a user message is introduced, any reasoning from previous turns are dropped.

For tool calling, we use XML-style special tags to reduce character escaping, following the observations of GLM-4.5 (Team, 2025a) and Qwen3-Coder (Yang et al., 2025a).

3.1.2. Data

Competition Math. For math, we use a similar strategy to Nemotron Nano 2 (NVIDIA, 2025d). However, we refresh the responses with GPT-OSS 120B (OpenAI, 2025). In addition, we create tool-integrated reasoning traces using Python tools and GPT-OSS 120B as the teacher model.

Competition Code. For code we use the same data from Nemotron Nano 2, which is made up of the prompts from Ahmad et al. (2025b) complemented with responses from DeepSeek-R1-0528 (DeepSeek-AI, 2025a).

Conversational Tool Use. We generate synthetic multi-turn trajectories to demonstrate conversational tool use. The generation of these trajectories involves a user that is given a task to accomplish, an agent that is instructed to help the user accomplish their task, and a tool execution environment, each of which is simulated by a language model. To limit the trajectories in the SFT training data to ones in which the actions of all of these entities are consistent with their goals, we employ a language model as a judge to evaluate the trajectories, and filter out trajectories for which the judge considers an action of an entity to be inconsistent with its goals. We use Qwen3-235B-A22B-Thinking-2507 (Yang et al., 2025a), Qwen3-32B (Yang et al., 2025a), GPT-OSS-120b (OpenAI, 2025), and Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025a) to generate data in this synthetic tool use trajectory generation pipeline.

Long Context. We generate synthetic data with a mean token length of 128k tokens and a maximum of 256k tokens to improve long-context performance, validated against a subset of RULER tasks.

Formal Proofs. For Lean theorem proving, we curated SFT data by first autoformalizating 580k natural language theorems from online mathematics communities (AoPS, Math StackExchange, MathOverflow) into 550k Lean 4 statements using an iterative refinement pipeline based on GPT-OSS-120B with backtranslation-based semantic verification. We then ran large-scale proof generation

using Goedel-Prover-V2-32B with up to 4 independent attempts and 8 self-correction rounds per statement, yielding 920k proof traces with compiler-verified solutions. After filtering, the final dataset contains 300k examples pairing formal theorem statements with successful reasoning traces and proofs.

Multilingual. We generate multilingual data in a similar manner to Nemotron Nano 2 (NVIDIA, 2025d). We used Qwen2.5-Instruct to translate our existing English post-training data into 5 target languages, French, Spanish, Italian, German and Japanese. Our pipeline translates inputs line-by-line and skips non-translatable content like code, XML tags, URLs, etc. Following the translation of the English source text, we utilized a language identification tool `https://pypi.org/project/langdetect/` to filter out samples that did not predominantly consist of target language tokens. Additionally, we excluded samples containing specific failure modes where the Qwen model explicitly stated its inability to translate the source text.

Our multilingual corpus was further comprised of 1.62 million text translation samples, aggregated from a combination of news-commentary datasets and proprietary sources. These samples covered bidirectional translation tasks between English and the five target languages.

Terminal Use. To teach Nemotron 3 Nano to complete autonomous terminal-based tasks, we generate a diverse set of verifiable tasks based on Terminal Bench (Team, 2025c). In particular, we adapt data from our competitive coding, competitive math, and long context datasets to terminal bench problems. We also constructed synthetic tasks requiring data analysis and file creation and operations. Additionally, we incorporated data from SWE-Smith (Yang et al., 2025b), which provides real-world software engineering tasks. We use Qwen3-Coder-480B-A35B-Instruct (Qwen, 2025) and Kimi-K2-Instruct-0905 (Team, 2025b) to generate action trajectories for each task using the Terminus-1 and Terminus-2 agents (Team, 2025c).

General Chat. We create SFT data by generating responses to the LMSYS (Zheng et al., 2023) and WildChat datasets (Li et al., 2024d) using GPT-OSS-120B, Qwen3-235B-A22B-Thinking-2507, and Qwen3-235B-A22B-Instruct-2507. The data is extended to multi-turn by having the same language model simulate the user and further continue the conversation.

Instruction Following. We create targeted instruction following data with the methodology used in Tülu 3 (Lambert et al., 2025). We simulate users in a conversation using language models seeded with a user persona from Nemotron-Personas-USA (Meyer & Corneil, 2025) and instructions from IFeval (Zhou et al., 2023) and IFBench (Pyatkin et al., 2025) train splits. The user language model is prompted to generate precise instruction following queries for one or more turns. We then use GPT-OSS-120B, Qwen3-235B-A22B-Thinking-2507, and Qwen3-235B-A22B-Instruct-2507 to generate responses to the user queries. The generated data is first filtered to only keep samples where all turns pass the respective instruction verifier implementations in IFEval and IFBench. Further filtering is done with a language model judge to remove samples where the responses only trivially or superficially follow instructions.

Safety. We compile a diverse set of unsafe prompts sourced from the Nemotron Content Safety v2 (Ghosh et al., 2025) and the Gretel Safety Alignment v1 (Gretel, 2024) datasets to target content safety risks, and Harmful Tasks (Hasan et al., 2024) and Red-Team-2K (Luo et al., 2024) datasets to target common jailbreak techniques. This collection is further balanced with safe prompts derived from Nemotron Content Safety v2.

For supervised fine-tuning (SFT), we apply safe prompt wrappers to unsafe prompts enabling the models to learn appropriate refusal behaviors while preserving user engagement. Various refusal strategies are implemented to align with good user experience. For instance, self-harm related prompts are paired with prompt templates encouraging the use of appropriate suicide prevention

helplines. A content-safety classifier is employed to filter the responses, ensuring alignment with safety objectives.

Software Engineering. To train Nemotron 3 Nano for autonomous software engineering capabilities including code exploration, issue reproduction and bug fixing, we curate a dataset of coding tasks derived from real-world GitHub issues. We use the issue description and containerized execution environments from SWE-Gym (Pan et al., 2025) and R2E-Gym (Jain et al., 2025) datasets. We distill trajectories from three open-source agent harnesses - OpenHands (Wang et al., 2025a), SWE-Agent (Yang et al., 2024), and Mini-SWE-Agent (Yang et al., 2024) using Qwen3-Coder-480B-A35B-Instruct (Qwen, 2025) as the teacher model.

Science. The science dataset spans physics, chemistry, and biology, and is produced through a unified pipeline that integrates synthetic, real, and document-based seed sources. We began by curating a set of challenging seed questions derived from Nemotron Nano v2 (NVIDIA, 2025d) as well as from scientific articles contained in the pre-training corpus. In parallel, we incorporated additional scientific articles from the same corpus as a complementary reservoir of seed material. Each article was annotated with three attributes: (1) education domain based on bert-based finetuned classifier (Li et al., 2024a), (2) content level (ranging from elementary to graduate), and (3) fine-grained topical categories (e.g., biology, chemistry, mathematics, law). Focusing on the graduate-level subset, we indexed these documents in a vector database and used a diverse set of science-oriented query prompts to retrieve thousands of highly relevant passages. These retrieved segments served as the foundation for generating multiple-choice question (MCQ) data, which were subsequently converted into an open-ended question-answering (OpenQA) format.

All seed sources—synthetic, real, and doc-retrieved—were subsequently processed through NeMo Data Designer (The NeMo Data Designer Team, 2025). The Data Designer was used to paraphrase prompts, produce multiple format and instruction variants, and enhance robustness across prompt styles. Reasoning traces for the SFT stage were generated using tool-integrated Python reasoning traces from GPT-OSS 120B (OpenAI, 2025). Crucially, all generated variants underwent rigorous LLM-judge filtering, ensuring strict format compliance, intent preservation, and high-quality reasoning consistency. During the RL stage, we further introduced targeted prompt and format augmentations to reduce prompt sensitivity and improve generalization.

A subset of STEM datasets developed in this work are released in both the multiple-choice question (MCQ[7]) and open question-answering (OpenQA[8]) formats to support Nano-V3 training and broader downstream research. These datasets are fully integrated into the RLVR pipeline, with both MCQ[9] and OpenQA[10] environments provided through NeMo Gym (NVIDIA, 2025b). This unified pipeline ensures consistent quality standards and supports robust reinforcement-learning-based evaluation and training across all STEM domains.

GenSelect. We improve our model's capability as a generative reward model by training it to identify the best solution among multiple candidates, following the approach in Toshniwal et al. (2025). We adapted the problems in our math and coding SFT data by generating synthetic solutions and then selection reasoning traces including their final verdicts using GPT-OSS 120B (OpenAI, 2025) and DeepSeek-R1-0528 (DeepSeek-AI, 2025a).

CUDA. We collect and synthesize 21k (PyTorch, Cuda C) pairs with seeds from HuggingFace Transformers (Wolf et al., 2020) and KernelBook (Paliskara & Saroufim, 2025). We first parse the

[7] https://huggingface.co/datasets/nvidia/Nemotron-RL-knowledge-mcqa
[8] https://huggingface.co/datasets/nvidia/Nemotron-RL-knowledge-openqa
[9] https://github.com/NVIDIA-NeMo/Gym/tree/main/resources_servers/mcqa
[10] https://github.com/NVIDIA-NeMo/Gym/tree/main/resources_servers/equivalence_llm_judge

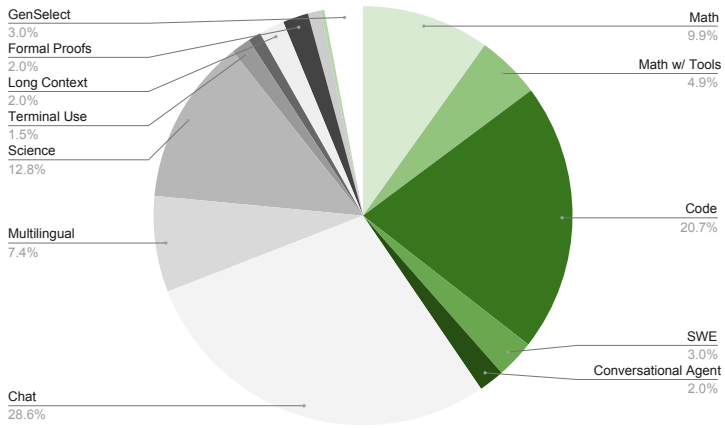


Figure 5 | SFT data blend for Nemotron 3 Nano.

PyTorch code from Transformers (Wolf et al., 2020) and KernelBook (Paliskara & Saroufim, 2025), and then use DeepSeek-R1-0528 (DeepSeek-AI, 2025a) to generate corresponding Cuda C code. We only include PyTorch, Cuda C pairs with Cuda C code that is successfully compiled and numerically verified against PyTorch reference code.

3.1.3. Data Filtering

For all domains, we apply a unified data filtering pipeline to ensure that only high-quality, license-compliant, and verifiable samples are used for training. We first discard malformed examples using structural checks (e.g., missing tool definitions when tool calls are present). We then aggressively filter reasoning traces exhibiting pathological repetition, such as repeated n-grams within a sliding window or across the entire trajectory, which we found to be a strong indicator of malformed or low-quality reasoning. Finally, based on internal audits of synthetically generated datasets, we observed that some teacher models occasionally produce reasoning traces and final responses that implicitly align with specific political entities or promote nationalistic narratives. To mitigate this, we apply targeted keyword- and regex-based filters (e.g., patterns such as "our nation/party [...]", "our values") and remove all trajectories matching such behavior.

3.1.4. Data Mixture

Our exact data blend can be found in Figure 5 (all datasets not listed make up less than 1% of the blend). We train over 18M total samples. For each dataset we decide how much data to include based on the approximate amount of data required to achieve optimal performance in single task settings. As the size of different datasets varies significantly, we employ a dynamic sampling approach where smaller datasets may be trained over for many epochs and larger datasets are trained for only a few epochs.

3.1.5. Reasoning Control

Nemotron 3 Nano allows for two different forms of reasoning control: reasoning on/off control and token budget control. Similar to NVIDIA (2025d), to enable reasoning on/off control we strip the reasoning traces from a random 10% of samples, and to enable budget control, we randomly truncate 3% of reasoning traces to different reasoning budgets, before continuing with the original post-reasoning response.

3.1.6. Hyperparameters

We train for 13000 steps using a batch size of 64 and employ sequence packing to a sequence length of 256K. We use a learning rate of $5 \cdot 10^{-5}$ and use 800 steps of learning rate warmup. We use a sequence-level MoE load balancing regularizer and set the loss coefficient to 10^{-4}.

3.2. Multi environment Reinforcement Learning from Verifiable Rewards

We employ a unified RLVR stage, training on all environments simultaneously. We find that this results in stable gains across all benchmarks throughout training, while single environment training often results in un-recoverable degradation of other benchmarks. We do two stages of such RLVR: one immediately after SFT and one after RLHF.

3.2.1. Environments

Competition Math. We train on the DAPO (Yu et al., 2025) and SkyWorks math (He et al., 2025) datasets. These datasets have 17K and 104K tasks respectively.

Competition Coding. We use competitive coding problems from Ahmad et al. (2025b). We limit the number of unit tests to 50 in order to reduce verification time. This filtering leaves us with 22K tasks.

Question Answering. We train on a variety of difficult multiple choice datasets focusing on STEM domains. Here the questions and answers are generated based on information from reference documents. This dataset has with 135K tasks.

Structured Outputs. We train Nemotron 3 Nano to have strong JSON schema adherence capabilities. We utilized NeMo Data Designer (The NeMo Data Designer Team, 2025) to create the seed dataset for RL. We start by constructing (JSON schema, document) pairs conditioned on diverse topics using Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025a). We then utilized these pairs to create RL prompts by taking the model to summarize the document according to the schema. To ensure high syntactic validity, the pipeline enforced strict complexity controls and applied rejection sampling, while simultaneously varying instruction difficulty and phrasing to maximize input diversity. This pipeline produces 9K tasks.

In the RL stage, a positive reward is given when the output matches the exact schema constraints, and no reward is given otherwise. For simplicity, we do not add a reward for the semantic content of the output.

Instruction Following. We use two instruction following environments during the training. The first environment is similar to the IFEval style environment used in NVIDIA (2025a), but with refreshed constraints from the IFBench training set (Pyatkin et al., 2025). We create 46K tasks for this environment.

The second environment uses LLM as a judge to verify whether or not the agent has followed complex instructions in multi-turn settings, where the instructions may be quite subtle. This environment is inspired by the Multi-Challenge benchmark (Deshpande et al., 2025). We create 3K total tasks for it.

Long Context. We generate challenging long-context QA pairs using Qwen3-235B-A22B-Thinking-2507 (Yang et al., 2025a), drawing from a subset of our pre-training mixture designed for multi-document synthesis. Each question is required to reference at least five documents, with the total input limited to 32k tokens. We employ Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025a) as the LLM judge to evaluate the model's rollouts. This dataset contains 12K tasks.

Agentic Tool Use. We use two environments to improve tool use capabilities. The first is Workplace Assistant, a multi-step verifiable tool-calling setup adapted from Styles (Styles et al., 2024) that was also used in Nemotron 2 Nano (NVIDIA, 2025d). This is a tool use - multi step agentic environment that tests the agent's ability to execute tasks in a workplace setting. Workplace Assistant contains a sandbox environment with five databases, 26 tools, and 690 tasks. These tasks represent common business activities, such as sending emails, scheduling meetings, etc. The correctness is verified through executing the tool calls issued by the agent and comparing it to the ground truth database state.

The second environment is a Multi-turn conversational agent environment. It tests an agent's tool-calling and proactive asking capability. Comprising approximately 1K tasks, this environment simulates complex banking scenarios like assisting customers with unblocking a credit card or solving account disputes. The correctness of the agent's actions is automatically verified by executing the tool calls it issues and comparing the resulting database state against the predefined ground truth.

3.2.2. Data Mixture and Curriculum

We begin by profiling all reinforcement learning (RL) tasks using our supervised fine-tuning (SFT) checkpoint. To focus training on challenging cases, we filter out samples where the SFT checkpoint already achieves a 100% pass rate. We then adopt the curriculum training method introduced in NVIDIA (2025a), which dynamically adjusts task difficulty throughout training.

In each batch, we maintain a fixed ratio of samples across different domains. For each domain, we model the target pass-rate distribution as a Gaussian function, shifting from high pass-rate (easier) samples early in training to low pass-rate (harder) samples later. The target mean of Gaussian distribution decreases linearly throughout training steps. Within each batch, samples from different domains are shuffled. This Gaussian sampling strategy prevents overfitting to either overly easy or overly difficult examples, ensuring a balanced learning progression.

This approach enables a controlled and gradual increase in task difficulty while preserving domain diversity and ensuring efficient batch composition. Figure 6 illustrates how sample difficulty evolves over the course of RL training. Once training progress plateaus, we re-profile the tasks using the best RL checkpoint and construct a new curriculum to further refine performance.

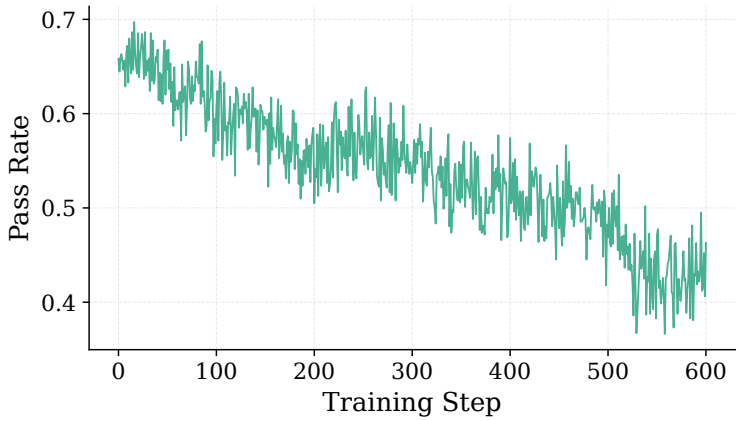


Figure 6 | Batch-wise pass rates across the RL curriculum.

We compare curriculum sampling against random sampling using an intermediate SFT checkpoint, maintaining identical domain ratios in both cases. As shown in Figure 7, curriculum sampling ensures stable learning across multiple domains throughout training. In contrast, random sampling

biases the model toward easier tasks, preventing it from effectively learning more challenging ones.

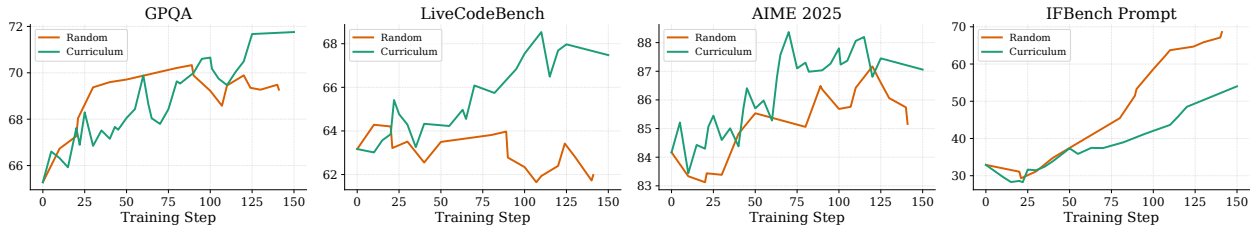


Figure 7 | Comparison between curriculum sampling and random sampling.

3.2.3. Surpassing SFT with RLVR

Recent works have demonstrated that supervised fine-tuning (SFT) alone on small models can achieve strong performance (Ahmad et al., 2025b; DeepSeek-AI, 2025a). In this study, we investigate whether RLVR can outperform a heavily fine-tuned SFT baseline. As illustrated in Figure 8, we compare the accuracy of model during RLVR training with two SFT checkpoints:

- SFT1: Our initial RLVR starting point, fine-tuned for approximately 3 epochs.
- SFT2: A heavily fine-tuned checkpoint, trained to full convergence (approximately 5 epochs).

Our results show that even with relatively short training, RLVR consistently exceeds or matches the accuracy of the heavily fine-tuned SFT model across all evaluated domains.

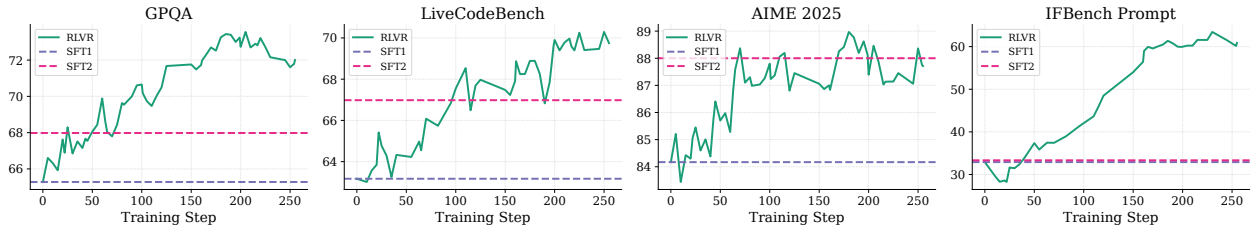


Figure 8 | RLVR surpasses or matches heavily fine-tuned SFT model across all evaluated domains.

3.2.4. Infrastructure

RL at the frontier of model post-training is currently defined by scaling up to an increasing diversity of tasks or environments designed for the model to learn increasingly general capabilities. Scaling RL to many environments requires a high-performance, extensible, and standardized interface for coordinating between rollouts and training. To address the scaling performance and extensibility challenges using one standard framework, we adopt NeMo Gym (NVIDIA, 2025b) and NeMo RL (NVIDIA, 2025c) for enabling large-scale RL on many different environments/verifiers.

NeMo Gym is based on the abstraction of *servers*. There are three core varieties of servers in Gym: (1) *agents*, (2) *models*, and (3) *resources*. An *agent* server implements the rollout kernel of a RL environment. A *model* server wraps an inference engine such as vLLM (Kwon et al., 2023) to provide a prompt-response API, and also carefully preserves token and inference log-prob data and metadata required for RL. A *resource* server provides a verification API for computing rewards from a given rollout.

Our Nemotron Nano 3 RLVR experiments were all based on an integrated infrastructure of NeMo RL and NeMo Gym: NeMo RL acts as the RL training loop controller, using Megatron-Core (Shoeybi et al., 2020) for model training at scale, and routing all rollouts through NeMo Gym and vLLM.

3.2.5. Algorithm

We train Nemotron 3 Nano using synchronous GRPO with masked importance sampling to mitigate training-inference misalignment (Shao et al., 2024; Team et al., 2025; Yao et al., 2025). We use 128 prompts per step and use 16 generations per prompt. We train with a batch size of 2048, making our updates on-policy. To further stabilize training we also freeze the MoE router weights. We employ the aux-loss-free load balancing approach and keep updating expert bias (Wang et al., 2024).

Our entire training run is done with a maximum generation length of 49K. We use overlong filtering (Yu et al., 2025), which we find boosts performance on reasoning intensive benchmarks.

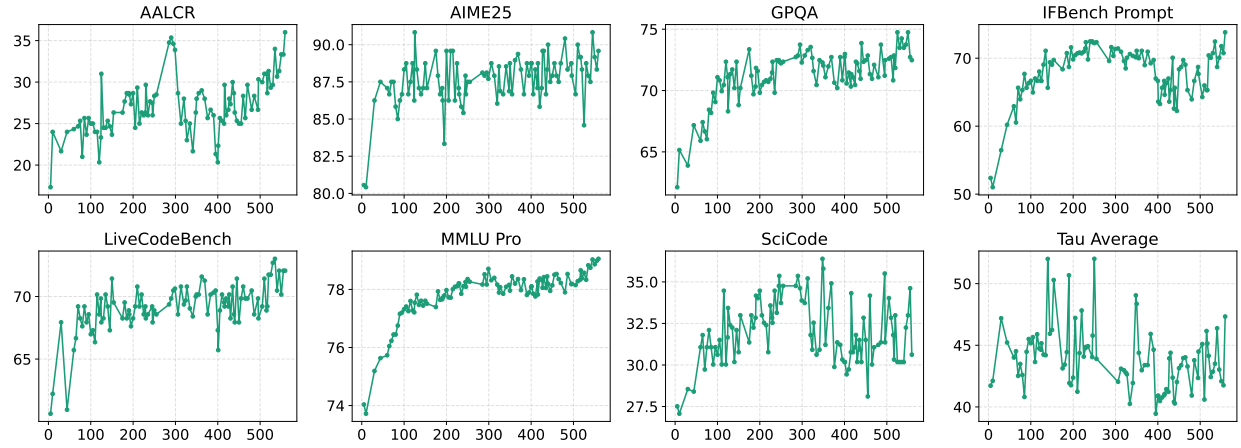


Figure 9 | Benchmark performance throughout RL training.

3.3. Reinforcement Learning from Human Feedback

3.3.1. Scaling Reinforcement Learning for Generative Reward-Model Training

Many recent works (Liu et al., 2025b; Wang et al., 2025b; Chen et al., 2025) have demonstrated that generative reward models (GenRMs) generalize better than traditional Bradley-Terry models, reducing the risk of reward hacking during RLHF. In order to train an accurate and robust GenRM, we leverage reinforcement learning at scale. Building on the methodology of Wang et al. (2025b), we train Qwen3-235B-A22B-Thinking-2507 (Yang et al., 2025a) to become a GenRM with GRPO algorithm. Given the conversation history, a new user request, and two candidate assistant responses, the GenRM first reasons through the strength and weakness of both responses, then produce an individual helpfulness score for each response as well as a ranking score. For GenRM training, we use 128 prompts per batch, 8 generations per prompt, and do one gradient step on the full batch. We define the reward as

$$\mathbf{R} = -C_1 I_{\text{format}} - |P_{h1} - G_{h1}| - |P_{h2} - G_{h2}| - C_2 \, |P_r - G_r| \,, \tag{1}$$

where P_r, G_r denote the predicted and ground-truth preference rankings; P_{h1}, G_{h1}, P_{h2}, G_{h2} denote the predicted and ground-truth helpfulness scores for responses 1 and 2, respectively; I_{format} indicates

whether the prediction violates the format requirement; C_1 and C_2 are hyper-parameters controlling the weights. We set $C_1 = 10$ and $C_2 = 1$.

We leverage data from HelpSteer3 (Wang et al., 2025b), a commercially-friendly subset of lmarena-ai/arena-human-preference-140k (Chiang et al., 2024), and a synthetic safety blend (see details in Appendix D) for model training. In our dataset, individual helpfulness scores range from 1 to 5, where higher means more helpful, while ranking score ranges from 1 to 6, in which 1 denotes that response 1 is far superior to response 2 and 6 denotes that response 2 is far superior to response 1 (Wang et al., 2025b). We augment each sample by switching positions of two responses to prevent positional bias. Figure 10 demonstrates that the performance of GenRM on RM-Bench (Liu et al., 2024), JudgeBench (Tan et al., 2024), and our internal validation set steadily improves as training progresses.

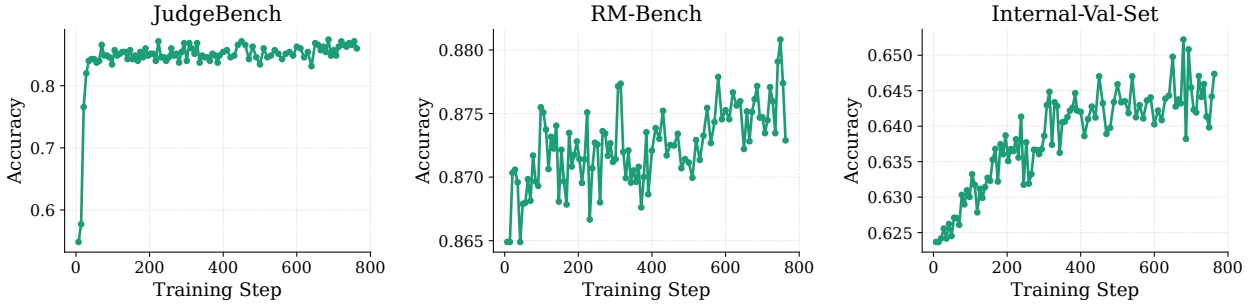


Figure 10 | GenRM performance improves across benchmarks as we scale up RL training.

3.3.2. RLHF with Group Relative Length Control

With a trained GenRM, we conduct RLHF on the same set of prompts. Same as RLVR, we use a batch of 128 prompts and 16 responses per prompt. Naively comparing all pairs of N responses would require $\binom{N}{2}$ GenRM calls per prompt, which scales quadratically and becomes prohibitively expensive for large N. With $N = 16$ responses, this would require 120 comparisons per prompt. Instead, we adopt a circular comparison strategy where each response is compared only with its successor: $(r_1, r_2), (r_2, r_3), \ldots, (r_{N-1}, r_N), (r_N, r_1)$, yielding exactly N comparisons. This reduces computational cost from $O(N^2)$ to $O(N)$ while still connecting all responses in a comparison graph. Each response is also judged twice in different positions so as to alleviate positional bias.

For each pairwise comparison (r_i, r_j), the GenRM produces individual helpfulness scores $s_i, s_j \in [1, 5]$ and a ranking score $s_r \in [1, 6]$. In the case where $s_i = s_j$, we further employ a simple tiebreaker mechanism:

$$s_i = s_i + (3.5 - s_r), \tag{2}$$

$$s_j = s_j + (s_r - 3.5). \tag{3}$$

The base reward $R_i^{(\text{base})}$ for response r_i is then computed by averaging its scores from two matches.

When training with base reward, we find that the length of response can rapidly increase as RLHF training proceeds. This is different from reward hacking, as the increase of length mostly comes from reasoning trace while only final answer is judged by GenRM. It is similar to observations in DeepSeek-AI (2025a) where model spends more inference time compute to achieve better rewards. However, unlike reasoning heavy tasks like math and coding, prompts in RLHF datasets usually don't

require extensive reasoning. In order to reduce redundant thinking, we propose a Group Relative Length Control mechanism during RLHF. Specifically, for each prompt, we generate a group of N candidate responses $\{r_1, r_2, \ldots, r_N\}$. Each response r_i is decomposed into a reasoning component $r_i^{(\text{think})}$ and an answer component $r_i^{(\text{answer})}$, with corresponding lengths $\ell_i^{(\text{think})}$ and $\ell_i^{(\text{answer})}$.

Length-Normalized Reward Adjustment. We compute a zero-mean, group-relative length bonus that encourages shorter responses within a group. For the reasoning component, we first normalize lengths within the group

$$w_i^{(\text{think})} = 1 - \frac{\ell_i^{(\text{think})} - \ell_{\min}^{(\text{think})}}{\ell_{\max}^{(\text{think})} - \ell_{\min}^{(\text{think})}}, \tag{4}$$

where $\ell_{\min}^{(\text{think})} = \min_j \ell_j^{(\text{think})}$ and $\ell_{\max}^{(\text{think})} = \max_j \ell_j^{(\text{think})}$. To ensure the adjustment is zero-sum across the group (preserving the overall reward scale), we center the weights

$$\tilde{w}_i^{(\text{think})} = w_i^{(\text{think})} - \frac{1}{N}\sum_{j=1}^{N} w_j^{(\text{think})}. \tag{5}$$

The same procedure is applied to answer lengths to obtain $\tilde{w}_i^{(\text{answer})}$. The final reward for response r_i is then

$$R_i = R_i^{(\text{base})} + \lambda^{(\text{think})}\tilde{w}_i^{(\text{think})} + \lambda^{(\text{answer})}\tilde{w}_i^{(\text{answer})}, \tag{6}$$

where $R_i^{(\text{base})}$ is the base reward from pairwise comparisons and $\lambda^{(\text{think})}, \lambda^{(\text{answer})}$ are coefficients controlling the strength of the length penalty. We set $\lambda^{(\text{think})} = 0.5$, $\lambda^{(\text{answer})} = 0.5$.

Quality-Gated Conciseness Bonus. To further encourage concise responses without sacrificing quality, we introduce optional bonuses for the shortest responses that achieve top-tier quality scores. Let τ_p denote the p-th percentile threshold of scores within the group. For the response r_k with minimum reasoning length:

$$R_k \leftarrow R_k + \beta^{(\text{think})} \cdot \mathbb{1}\left[R_k^{(\text{base})} \geq \tau_p\right]$$

Similarly, for the response r_m with minimum answer length:

$$R_m \leftarrow R_m + \beta^{(\text{answer})} \cdot \mathbb{1}\left[R_m^{(\text{base})} \geq \tau_p\right]$$

where $\beta^{(\text{think})}$ and $\beta^{(\text{answer})}$ are the reasoning and answer conciseness bonuses respectively, and $\mathbb{1}[\cdot]$ is the indicator function. We set $\beta^{(\text{think})} = 0.5$, $\beta^{(\text{answer})} = 0.5$, and $\tau_p = 80$.

This mechanism ensures that (1) length penalties are relative within each prompt group rather than absolute, avoiding bias against inherently complex problems; and (2) conciseness bonuses are only awarded to high-quality responses, preventing the model from learning to produce short but low-quality answers. We observe that the verbosity level reduces 30% during the training without sacrificing accuracy.

Benchmark	N-3-Nano	Qwen3	GPT-OSS
General Knowledge			
MMLU-Pro	78.30	**80.90**	75.00
Reasoning			
AIME25 (no tools)	89.06	85.00	**91.70**
AIME25 (with tools)	**99.17**	-	98.7
GPQA (no tools)	73.04	**73.40**	71.50
GPQA (with tools)	**75.00**	-	74.20
LiveCodeBench (v6 2024-08↔2025-05)	**68.25**	66.00	61.00
SciCode (subtask)	33.28	33.00	**34.00**
HLE (no tools)	10.57	9.80	**10.90**
HLE (with tools)	15.48	-	**17.30**
MiniF2F pass@1	**50.03**	5.72*	12.05*
MiniF2F pass@32	**79.92**	16.80*	43.03*
Agentic			
Terminal Bench (hard subset)	8.51	5.00	**10.00**
SWE-Bench (OpenHands)	**38.76**	22.00*	34.00*
TauBench V2			
Airline	48.00	**58.00**	38.00
Retail	56.91	**58.80**	54.80
Telecom	42.21	26.30	**49.70**
Average	**49.04**	47.70	47.50
BFCL v4	**53.76**	46.40*	-
Chat & Instruction Following			
IFBench (prompt)	**71.51**	51.00	65.00
Scale AI Multi Challenge	38.45	**44.75**	33.75
Arena-Hard-V2 (Hard Prompt)	**72.10**	49.60*	71.20*
Arena-Hard-V2 (Creative Writing)	63.20	**66.00***	25.90*
Arena-Hard-V2 (Average)	**67.65**	57.80	48.55
Long Context			
AA-LCR	35.85	**59.00**	34.00
RULER-100 @ 256k	**92.92**	89.40	-
RULER-100 @ 512K	**91.25**	84.00	-
RULER-100 @ 1M	**86.34**	77.50	-
Multilingual			
MMLU-ProX (avg over langs)	59.50	**77.60***	69.10*
WMT24++ (en→xx)	**86.20**	85.60	83.20

Table 3 | Nemotron 3 Nano compared to Qwen3-30B-A3B-Thinking-2507, and GPT-OSS 20B.

3.4. Post-trained Model Evaluations

3.4.1. Evaluation Benchmarks

We evaluate Nemotron 3 Nano across a broad suite of established benchmarks spanning mathematical and scientific reasoning, coding, agentic tool use, instruction following, long-context understanding, and multilingual capability. Table 3 summarizes the final results.

All evaluation results were collected via Nemo Evaluator SDK[11] and for most benchmarks, the Nemo Skills Harness[12]. For reproducibility purposes, the open source container on Nemo Skills packaged via NVIDIA's Nemo Evaluator SDK used for evaluations can be found here[13]. In addition to Nemo Skills, the evaluations also used dedicated packaged containers for Tau-2 Bench, ArenaHard v2, AA_LCR. More details on the evaluation settings can be found in the Nemo Evaluator SDK configs folder[14]. The following benchmarks are not onboarded yet in our open source tools and for these we used their official open source implementation: Terminal Bench, SWE-Bench, Scale AI Multi Challenge.

For mathematical and STEM reasoning, we evaluate on AIME25 (with and without tools), GPQA (Rein et al., 2023), LiveCodeBench v6 (Jain et al., 2024), SciCode (Tian et al., 2024), and Humanity's Last Exam (Phan et al., 2025). We additionally include MMLU-Pro to assess general academic and knowledge-intensive reasoning.

Agentic and tool-augmented capabilities are measured using TerminalBench, SWE-Bench (OpenHands) (Jimenez et al., 2023; Wang et al., 2025a), TauBench V2 (airline, retail, telecom) (Barres et al., 2025), and BFCL v4 (Patil et al., 2025), each of which provides verifiable reward signals via unit tests, database state transitions, or structured schema constraints.

Instruction-following and conversational ability are evaluated with IFBench, Scale AI Multi-Challenge, and Arena-Hard-V2 (Li et al., 2024c). These benchmarks probe multi-constraint instructions, preference-aligned chat behavior, and faithfulness to user intent. For Arena-Hard-V2, we follow Yang et al. (2025a) and use GPT-4.1 as judge.

Long-context performance is assessed with RULER-100 at 256k, 512k, and 1M tokens (Hsieh et al., 2024), together with AA-LCR, evaluating retrieval, stability, and chain-of-thought coherence over extreme context lengths. RULER-100 is evaluated with reasoning off, whereas AA-LCR is measured with reasoning on.

For multilingual capability, we report results on MMLU-ProX (Xuan et al., 2025) and WMT24++ (en→xx) (Deutsch et al., 2025), covering a mix of reasoning and translation settings across multiple high-resource languages.

For comparison with GPT-OSS 20B and Qwen3-30B-A3B-Thinking-2507, we use the officially reported numbers whenever available; if a benchmark is not reported, we take the value from ArtificialAnalysis (AA)[15]; and if neither source provides results, we may compute the scores ourselves using the official evaluation protocol.

Table 3 presents a comprehensive performance comparison between the three models. Nemotron 3 Nano shows strong results, surpassing both GPT-OSS 20B and Qwen3-30B-A3B-Thinking-2507 in all categories. On reasoning benchmarks Nemotron 3 Nano surpasses the Qwen3 model and is

[11] https://github.com/NVIDIA-NeMo/Evaluator
[12] https://github.com/NVIDIA-NeMo/Skills
[13] https://catalog.ngc.nvidia.com/orgs/nvidia/teams/eval-factory/containers/nemo_skills
[14] https://github.com/NVIDIA-NeMo/Evaluator
[15] https://artificialanalysis.ai/

competitive with GPT-OSS, which was the previous best model in these categories. In the agentic, chat, and long context categories Nemotron 3 Nano significantly outperforms both of the other models, demonstrating the strength of our post-training pipeline.

4. Quantization

After post-training the model in BF16, we applied Post-Training Quantization (PTQ) using ModelOpt[16] and Megatron-LM to quantize the model to FP8.

4.1. Post-Training Quantization Calibration Dataset

For PTQ calibration, we used a small subset containing 1K samples from post-training reasoning SFT dataset. Using calibration data based on the post-training SFT data yielded slightly better accuracy recovery compared to the `cnn_dailymail` dataset.

We also ablated with PTQ using calibration data curated from on-policy generations from the BF16 model, but did not observe any benefit in accuracy recovery compared to the SFT-based calibration dataset.

4.2. Selective Post-Training Quantization

To preserve accuracy while improving efficiency, we used a selective quantization strategy. We performed quantization sensitivity analysis and explored a set of quantization configurations for mixed-precision models. This study showed that self-attention layers (6 out of 52 layers for Nemotron 3 Nano) are the most sensitive components, hence we keep them in BF16. Also, the Mamba layers that feed into the self-attention layers were found to be sensitive and are kept in BF16. Overall, keeping the 6 self-attention layers and the 6 Mamba layers in BF16 provided a sweet-spot configuration for accuracy recovery and efficiency trade-off.

The model weights, activations, and KV cache are quantized to FP8. Conv1D within all the Mamba layers are kept in BF16.

4.3. Accuracy and Throughput

Table 4 compares accuracy numbers of Nemotron 3 Nano FP8 with BF16 on multiple benchmarks. Overall, the FP8 model achieves approximately 99% median accuracy recovery compared to the BF16 model.

To verify the effectiveness of our selective quantization strategy and to better understand the accuracy–efficiency trade-off, we evaluated several quantization configurations. We conducted an ablation study by applying PTQ to different model components. Specifically, we examined three factors: attention layer quantization (BF16 or FP8), Mamba layer quantization (FP8 or a mix of BF16 and FP8), and KV cache quantization (BF16 or FP8).

As shown in Figure 11, KV cache with FP8 quantization significantly improves throughput by enabling larger batch sizes. While other quantization configurations suffer from accuracy degradation, our selective quantization can retain the accuracy numbers even with KV cache quantization. The results confirm that retaining the self-attention layers and their preceding Mamba layers in BF16, while quantizing the remaining layers and the KV cache in FP8, yields a strong accuracy–efficiency trade-off.

[16] https://github.com/NVIDIA/Model-Optimizer

Benchmark	N-3-Nano BF16	N-3-Nano FP8
General Knowledge		
MMLU-Pro	78.30	78.10
Reasoning		
AIME25 (no tools)	89.06	87.71
AIME25 (with tools)	99.17	98.80
GPQA (no tools)	73.04	72.47
GPQA (with tools)	75.00	73.40
LiveCodeBench (v6 2024-08↔2025-05)	68.25	67.62
SciCode (subtask)	33.28	31.88
HLE (no tools)	10.57	10.33
HLE (with tools)	15.48	14.27
Agentic		
TauBench V2		
Airline	48.00	44.79
Retail	56.91	55.59
Telecom	42.21	40.75
Average	49.04	47.04
BFCL v4	53.76	53.15
Chat & Instruction Following		
IFBench (prompt)	71.51	72.19
Long Context		
AA-LCR	35.85	36.06
Multilingual		
MMLU-ProX (avg over langs)	59.50	59.63

Table 4 | Accuracy numbers of Nemotron 3 Nano before/after FP8 quantization.

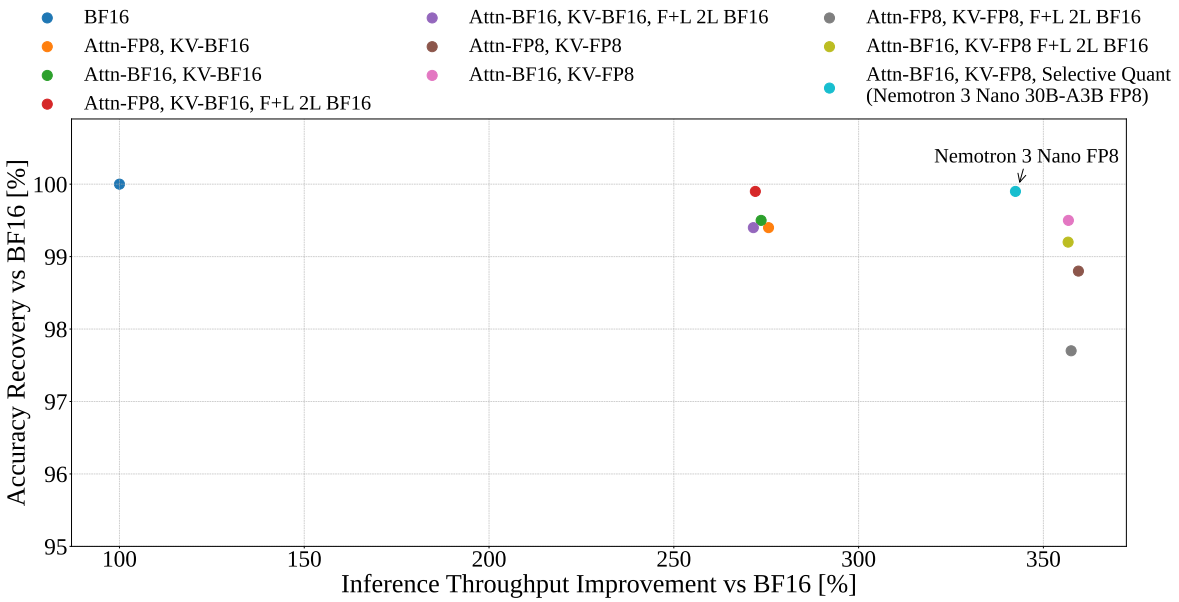


Figure 11 | Ablation study of different quantization configurations for accuracy–throughput trade-offs. Accuracy recovery and throughput improvements are computed relative to the Nemotron 3 Nano BF16 checkpoint, with values normalized such that the BF16 baseline is 100%. Accuracy recovery is defined as the median of the recovery rates across all benchmarks. The benchmark was conducted on a single H100 with ISL/OSL=8K/16K. Given that more aggressively quantized models can accommodate larger batch sizes due to lower memory footprint, we used the maximum batch size for each quantization configuration for fair comparisons under the same hardware constraints.

5. Conclusion

We present Nemotron 3 Nano, an open and efficient MoE Hybrid Mamba-Transformer model for agentic reasoning. Nemotron 3 Nano achieves better or on-par accuracy than competitive models while having up-to 3.3× higher inference throughput. Nemotron 3 Nano supports context lengths of up to 1M tokens. We have released the weights for both the base (Nemotron 3 Nano 30B-A3B Base) and final (Nemotron 3 Nano 30B-A3B) models on HuggingFace. Along with the weights, we have also open-sourced the training recipe, data, and code.

Contributors

We thank the following people for their invaluable contributions to NVIDIA Nemotron 3 Nano.

Pretraining Data. Abhinav Khattar, Aleksander Ficek, Alisa Liu, Arham Mehta, Asif Ahamed, Ayush Dattagupta, Benedikt Schifferer, Brandon Norick, Branislav Kisacanin, Dan Su, Dane Corneil, Daria Gitman, Dhruv Nathawani, Dima Rekesh, Divyanshu Kakwani, Edgar Minasyan, Eileen Long, Ellie Evans, Eric Tramel, Evelina Bakhturina, Felipe Soares, Feng Chen, Gantavya Bhatt, George Armstrong, Igor Gitman, Ivan Moshkov, Jane Polak Scowcroft, John Kamalu, Johnny Greco, Joseph Jennings, Jupinder Parmar, Kezhi Kong, Markus Kliegl, Maarten Van Segbroeck, Matvei Novikov, Mehrzad Samadi, Miguel Martinez, Mohammad Shoeybi, Mostofa Patwary, Nabin Mulepati, Oleksii Hrinchuk, Rabeeh Karimi Mahabadi, Rima Shahbazyan, Riyad Islam, Roger Waleffe, Rohit Watve, Sadegh Mahdavi, Sanjeev Satheesh, Sean Narentharen, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shuoyang Ding, Somshubra Majumdar, Stephen Ge, Sumeet Kumar Barua,

Suseella Panguluri, Syeda Nahida Akter, Vahid Noorozi, Vitaly Kurin, Vitaly Lavrukhin, Wasi Uddin Ahmad, Wei Du, Wei Ping, Yejin Choi, Yev Meyer, Ying Lin, Zihan Liu

Architecture. Abhinav Khattar, Bita Darvish Rouhani, Deepak Narayanan, Ilya Loshchilov, Jatin Mitra, Joey Guman, Mohammad Shoeybi, Mostofa Patwary, Kezhi Kong, Krishna C. Puvvada, Maor Ashkenazi, Nidhi Bhatia, Pavlo Molchanov, Rabeeh Karimi Mahabadi, Rasoul Shafipour, Ritika Borkar, Roger Waleffe, Ryan Prenger, Sanjeev Satheesh, Venmugil Elango, Yonggan Fu

Pretraining Software. Aarti Basant, Ashwath Aithal, Abhinav Khattar, Deepak Narayanan, Duncan Riach, Eric Harper, Hexin Wang, Jared Casper, Jimmy Zhang, Kezhi Kong, Mike Chrzanowski, Nima Tajbakhsh, Pranav Prashant Thombre, Roger Waleffe, Russell J. Hewett, Seonmyeong Bak, Shiqing Fan, Vijay Korthikanti, Xiaowei Ren, Yashaswi Karnati, Zijie Yan

Pretraining. Abhinav Khattar, Brandon Norick, Dan Su, Eric Tramel, Deepak Narayanan, John Kamalu, Joseph Jennings, Jupinder Parmar, Markus Kliegl, Miguel Martinez, Mohammad Shoeybi, Mostofa Patwary, Kezhi Kong, Kevin Shih, Rabeeh Karimi Mahabadi, Roger Waleffe, Ryan Prenger, Shrimai Prabhumoye, Sanjeev Satheesh, Syeda Nahida Akter, Ying Lin

Long Context. Boris Ginsburg, Cheng-Ping Hsieh, Dan Su, Dima Rekesh, Faisal Ladhak, Fei Jia, John Kamalu, Kezhi Kong, Krishna C. Puvvada, Markus Kliegl, Mostofa Patwary, Roger Waleffe, Samuel Kriman, Sanjeev Satheesh, Shantanu Acharya, Simeng Sun, Ushnish De

Posttraining Software. Adi Renduchintala, Alexander Bukharin, Ali Taghibakhshi, Banghua Zhu, Brian Yu, Duncan Riach, Frankie Siino, Gerald Shen, Jiaqi Zeng, Kezhi Kong, Li Ding, Luis Vega, Maanu Grover, Marc Romeijn, Parth Chadha, Peter Jin, Soumye Singhal, Terry Kong, Tugrul Konuk, Yi-Fu Wu, Yubo Gao

Posttraining. Abhibha Gupta, Adi Renduchintala, Akanksha Shukla, Aleksander Ficek, Alexander Bukharin, Ameya Sunil Mahabaleshwarkar, Banghua Zhu, Besmira Nushi, Branislav Kisacanin, Cheng-Ping Hsieh, Charles Wang, Damon Mosk-Aoyama, Daria Gitman, Dhruv Nathawani, Dima Rekesh, Edgar Minasyan, Edward Lin, Evelina Bakhturina, Fei Jia, Felipe Soares, Feng Chen, George Armstrong, Grigor Nalbandyan, Haifeng Qian, Hayley Ross, Igor Gitman, Ivan Moshkov, Jeffrey Glick, Jiaqi Zeng, Jian Zhang, Jie Lou, Julien Veron Vialard, Junkeun Yi, Katherine Luna, Khushi Bhardwaj, Krishna C. Puvvada, Luis Vega, Makesh Narsimhan Sreedhar, Matvei Novikov, Mehrzad Samadi, Mengru Wang, Michael Evans, Nikolai Ludwig, Oleksii Hrinchuk, Oleksii Kuchaiev, Olivier Delalleau, Ouye Xie, Peter Jin, Pritam Gundecha, Prasoon Varshney, Rima Shahbazyan, Ritu Gala, Sadegh Mahdavi, Sahil Modi, Sanjay Kariyappa, Sean Narenthiran, Shantanu Acharya, Shubham Toshniwal, Shuoyang Ding, Somshubra Majumdar, Soumye Singhal, Stephen Ge, Sugam Dipak Devare, Suseella Panguluri, Tugrul Konuk, Vahid Noroozi, Venkat Srinivasan, Vitaly Lavrukhin, Wasi Uddin Ahmad, Wei Du, Yev Meyer, Yian Zhang, Yoshi Suhara

Evaluation, Safety and Release. Aaron Grattafiori, Barnaby Simkin, Besmira Nushi, Bilal Kartal, Christopher Parisien, Daniel Rohrer, David Mosallanezhad, Eileen Peters Long, Erick Galinkin, Fay Wang, Ferenc Galko, Gorkem Batmaz, Jane Polak Scowcroft, Katherine Luna, Khushi Bhardwaj, Leon Derczynski, Michael Boone, Michael Evans, Piotr Januszewski, Rich Harang, Rishabh Garg, Riyad Islam, Sanjay Kariyappa, Sanjeev Satheesh, Shaona Ghosh, Wojciech Prazuch, Yoshi Subara, Zhen Dong, Zijia Chen

Infrastructure. Aaron Blakeman, Anubhav Mandarwal, Alex Kondratenko, Aleksandr Shaposhnikov, Ashwin Poojary, Brandon Soubasis, Collin Neale, Dong Ahn, Evan Briones, Gargi Prasad, Harsh Sharma, Herman Sahota, Himanshu Soni, Jining Huang, Kumar Anik, Maer Rodrigues de Melo, Nikhil Jukar, Pasha Shamis, Rick Izzo, Ruoxi Zhang, Satish Pasumarthi, Sergey Kashirsky, Shelby Thomas, Stefania Alborghetti

Quantization. Aditya Vavre, Akhiad Bercovich, Ameya Sunil Mahabaleshwarkar, Amnon Geifman, Asma Kuriparambil Thekkumpate, Ben Lanir, Bilal Kartal, Chenhan Yu, Daniel Afrimi, Darko Stosic, Dusan Stosic, Ganesh Ajjanagadde, Huizi Mao, Ido Shahaf, Jenny Chen, Kai Xu, Nave Assaf, Omer Ullman Argov, Ran Zilberstein, Sharath Turuvekere Sreenivas, Sweta Priyadarshi, Tijmen Blankevoort, Tomer Asida, Yoshi Suhara, Zach Moshe, Zijia Chen

Inference. Amir Klein, Amit Zuker, Chenghao Zhang, Daniel Afrimi, Daniel Serebrenik, Gal Hubara Agam, Helen Ngo, Joyjit Daw, Kan Zhu, Keshav Santhanam, Lawrence McAfee, Lucas Liebenwein, Luis Vega, Nave Assaf, Neta Zmora, Netanel Haber, Omer Ullman Argov, Peter Dykas, Pranav Prashant Thombre, Ran Zilberstein, Roi Koren, Shahar Mor, Shanmugam Ramasamy, Siddharth Singh, Suyog Gupta, Teodor-Dumitru Ene, Tomer Asida, Tomer Bar Natan, Vijay Korthikanti, Wanli Jiang, William Zhang, Yashaswi Karnati

Deployment. Alexandre Milesi, Anahita Bhiwandiwalla, Huy C Nguyen, Huy Q Nguyen, Izzy Putterman, Manoj Kilaru, Maryam Moosaei, Pawel Morkisz, Tan Bui, Thanh Do

Legal and Compliance. Barnaby Simkin, Chantal Hwang, Chetan Mungekar, Dina Yared, Hiren Upadhyay, Iain Cunningham, Katherine Cheung, Laya Sleiman, Meredith Price, Michael Boone, Nikki Pope, Saori Kaji

Marketing. Amelia Barton, Chintan Patel, Erik Pounds, Mark Cai, Natalie Hereth, Nicola Sessions, Nirmal Juluru, Shreya Gopal, Will Jennings

Project Management. Amy Shen, Ann Guan, Bardiya Sadeghi, Daria Levy, Elena Lantz, Elliott Ning, Krzysztof Pawelec, Melissa Corpuz, Negar Habibi, Pinky Xu, Qing Miao, Ryan Timbrook, Seth Poulos, Smita Ithape, Twinkle Vashishth

Product. Chris Alexiuk, Ellie Evans, Jane Polak Scowcroft, Jesse Oliver, Joey Conway, Tom Balough, Udi Karpas, Wenfei Zhou

Leadership. Andrew Tao, Bita Darvish Rouhani, Boris Ginsburg, Bryan Catanzaro, Carlo del Mundo, Eileen Long, Eric Chung, Jane Polak Scowcroft, Jan Kautz, Jian Zhang, Joey Conway, Jonathan Cohen, Kari Briski, Mohammad Shoeybi, Mostofa Patwary, Oleksii Kuchaiev, Oluwatobi Olabiyi, Pavlo Molchanov, Ran El-Yaniv, Ran Zilberstein, Yonatan Geifman, Yejin Choi

References

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. OpenCodeInstruct: A Large-scale Instruction Tuning Dataset for Code LLMs. *arXiv preprint arXiv:2504.04030*, 2025a.

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. *arXiv preprint arXiv:2504.01943*, 2025b.

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, 2023. URL https://arxiv.org/abs/2305.13245.

Syeda Nahida Akter, Shrimai Prabhumoye, Eric Nyberg, Mostofa Patwary, Mohammad Shoeybi, Yejin Choi, and Bryan Catanzaro. Front-Loading Reasoning: The Synergy Between Pretraining and Post-Training Data. *arXiv preprint arXiv:2510.03264*, 2025.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv preprint arXiv:2506.07982*, 2025.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. RM-R1: Reward Modeling as Reasoning. *arXiv preprint arXiv:2505.02387*, 2025.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference, 2024.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. *arXiv preprint arXiv:2401.06066*, 2024.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality, 2024. URL https://arxiv.org/abs/2405.21060.

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025a. URL https://arxiv.org/abs/2501.12948.

DeepSeek-AI. DeepSeek-V3 Technical Report, 2025b. URL https://arxiv.org/abs/2412.19437.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. MultiChallenge: A Realistic Multi-Turn Conversation Evaluation Benchmark Challenging to Frontier LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18632–18702, 2025.

Daniel Deutsch, Eleftheria Briakou, Isaac Rayburn Caswell, Mara Finkelstein, Rebecca Galor, Juraj Juraska, Geza Kovacs, Alison Lui, Ricardo Rei, Jason Riesa, et al. WMT24++: Expanding the Language Coverage of WMT24 to 55 Languages & Dialects. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12257–12284, 2025.

Steven Feng, Shrimai Prabhumoye, Kezhi Kong, Dan Su, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Maximize Your Data's Potential: Enhancing LLM Accuracy with Two-Phase Pretraining, 2024. URL https://arxiv.org/abs/2412.15285.

Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Ohi, Masaki Kawamura, Taishi Nakamura, Takumi Okamoto, et al. Rewriting Pre-Training Data Boosts LLM Performance in Math and Code. *arXiv preprint arXiv:2505.02881*, 2025. URL https://arxiv.org/abs/2505.02881.

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. AEGIS2.0: A Diverse AI Safety Dataset and Risks Taxonomy for Alignment of LLM Guardrails. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5992–6026, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.306. URL https://aclanthology.org/2025.naacl-long.306/.

Gretel. Gretel Synthetic Safety Alignment Dataset, 12 2024. URL https://huggingface.co/datasets/gretelai/gretel-safety-alignment-en-v1.

Adib Hasan, Ileana Rugina, and Alex Wang. Pruning for Protection: Increasing Jailbreak Resistance in Aligned LLMs Without Fine-Tuning. *arXiv preprint arXiv:2401.10862*, 2024.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, et al. Skywork Open Reasoner 1 Technical Report. *arXiv preprint arXiv:2505.22312*, 2025.

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*, 2021.

Andrew Hojel, Michael Pust, Tim Romanski, Yash Vanjani, Ritvik Kapila, Mohit Parmar, Adarsh Chaluvaraju, Alok Tripathy, Anil Thomas, Ashish Tanwer, Darsh J Shah, Ishaan Shah, Karl Stratos, Khoi Nguyen, Kurt Smith, Michael Callahan, Peter Rushton, Philip Monk, Platon Mazarakis, Saad Jamal, Saurabh Srivastava, Somanshu Singla, and Ashish Vaswani. Essential-Web v1.0: 24T tokens of organized web data, 2025. URL https://arxiv.org/abs/2506.14111.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models? *arXiv preprint arXiv:2404.06654*, 2024.

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, dahai li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J Yang, JH Liu, Chenchen Zhang, Linzheng Chai, et al. Opencoder: The open cookbook for top-tier code large language models. *arXiv preprint arXiv:2411.04905*, 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. *arXiv preprint arXiv:2403.07974*, 2024.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural Environments and Hybrid Verifiers for Scaling Open-Weights SWE Agents, 2025. URL https://arxiv.org/abs/2504.07164.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *arXiv preprint arXiv:2310.06770*, 2023.

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.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. DataComp-LM: In Search of the Next Generation of Training Sets for Language Models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024a.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024b.

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, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024c.

Xuehai Li, Zi Ye, Xiaoxin Zhang, Xinshi Lu, Yingqiang Xia, Bairu Wu, Shihan Dong, Qipeng Jin, Jialu Wang, Heng Ji, et al. WildChat: 1M ChatGPT Interaction Logs in the Wild. *arXiv preprint arXiv:2405.01470*, 2024d.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A Hybrid Transformer-Mamba Language Model, 2024. URL https://arxiv.org/abs/2403.19887.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. RM-Bench: Benchmarking Reward Models of Language Models with Subtlety and Style . *arXiv preprint arXiv:2410.16184*, 2024.

Zihan Liu, Zhuolin Yang, Yang Chen, Chankyu Lee, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. AceReason-Nemotron 1.1: Advancing Math and Code Reasoning through SFT and RL Synergy. *arXiv preprint arXiv:2506.13284*, 2025a.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. JailBreakV: A Benchmark for Assessing the Robustness of MultiModal Large Language Models against Jailbreak Attacks, 2024. URL https://arxiv.org/abs/2404.03027.

Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC-Math: A 133 Billion-Token-Scale High Quality Math Pretraining Dataset, 2025. URL https://arxiv.org/abs/2508.15096.

Somshubra Majumdar, Vahid Noroozi, Mehrzad Samadi, Sean Narenthiran, Aleksander Ficek, Wasi Uddin Ahmad, Jocelyn Huang, Jagadeesh Balam, and Boris Ginsburg. Genetic Instruct: Scaling up Synthetic Generation of Coding Instructions for Large Language Models. *arXiv preprint arXiv:2407.21077*, 2024.

Yev Meyer and Dane Corneil. Nemotron-Personas-USA: Synthetic personas aligned to real-world distributions, June 2025. URL https://huggingface.co/datasets/nvidia/Nemotron-Personas-USA.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*, 2024.

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. *arXiv preprint arXiv:2504.16891*, 2025.

Grigor Nalbandyan, Rima Shahbazyan, and Evelina Bakhturina. SCORE: Systematic COnsistency and Robustness Evaluation for Large Language Models. *arXiv preprint arXiv:2503.00137*, 2025.

Tue Nguyen. IChO-IPhO-RL-v2-formated, 2025. URL https://huggingface.co/datasets/II-Vietnam/IChO-IPhO-RL-v2-formated.

NVIDIA. Llama-Nemotron: Efficient Reasoning Models, 2025a. URL https://arxiv.org/abs/2505.00949.

NVIDIA. NeMo Gym: An Open Source Framework for Scaling Reinforcement Learning Environments for LLM. https://github.com/NVIDIA-NeMo/Gym, 2025b. GitHub repository.

NVIDIA. NeMo RL: A Scalable and Efficient Post-Training Library. https://github.com/NVIDIA-NeMo/RL, 2025c. GitHub repository.

NVIDIA. NVIDIA Nemotron Nano 2: An Accurate and Efficient Hybrid Mamba-Transformer Reasoning Model. *arXiv preprint arXiv:2508.14444*, 2025d.

NVIDIA. Nemotron-H: A Family of Accurate and Efficient Hybrid Mamba-Transformer Models, 2025e. URL https://arxiv.org/abs/2504.03624.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Sahan Paliskara and Mark Saroufim. Kernelbook, 5 2025. URL https://huggingface.co/datasets/GPUMODE/KernelBook.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training Software Engineering Agents and Verifiers with SWE-Gym, 2025. URL https://arxiv.org/abs/2412.21139.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Forty-second International Conference on Machine Learning*, 2025.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam, 2025. URL https://arxiv.org/abs/2501.14249.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*, 2025.

Qwen. Qwen2.5 Technical Report, 2025. URL https://arxiv.org/abs/2412.15115.

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.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *arXiv preprint arXiv:1701.06538*, 2017.

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.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. Workbench: a benchmark dataset for agents in a realistic workplace setting. *arXiv preprint arXiv:2405.00823*, 2024. doi: 10.48550/arXiv.2405.00823.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pp. 2459–2475, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.123. URL https://aclanthology.org/2025.acl-long.123/.

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. *arXiv preprint arXiv:2410.12784*, 2024.

GLM-4.5 Team. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models, 2025a. URL https://arxiv.org/abs/2508.06471.

Kimi Team. Kimi K2: Open Agentic Intelligence, 2025b. URL https://arxiv.org/abs/2507.20534.

Ling Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, et al. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model. *arXiv preprint arXiv:2510.18855*, 2025.

The Terminal-Bench Team. Terminal-bench: A benchmark for ai agents in terminal environments, Apr 2025c. URL https://github.com/laude-institute/terminal-bench.

NVIDIA The NeMo Data Designer Team. Nemo data designer: A framework for generating synthetic data from scratch or based on your own seed data. https://github.com/NVIDIA-NeMo/DataDesigner, 2025. GitHub Repository.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. SciCode: A Research Coding Benchmark Curated by Scientists, 2024. URL https://arxiv.org/abs/2407.13168.

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. *arXiv preprint arXiv:2410.01560*, 2024.

Shubham Toshniwal, Ivan Sorokin, Aleksander Ficek, Ivan Moshkov, and Igor Gitman. GenSelect: A Generative Approach to Best-of-N, 2025. URL https://arxiv.org/abs/2507.17797.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-Loss-Free Load Balancing Strategy for Mixture-of-Experts. *arXiv preprint arXiv:2408.15664*, 2024.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents, 2025a. URL https://arxiv.org/abs/2407.16741.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. HelpSteer3-Preference: Open Human-Annotated Preference Data across Diverse Tasks and Languages. *arXiv preprint arXiv:2505.11475*, 2025b.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick

von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Weihao Xuan, Rui Yang, Heli Qi, Qingcheng Zeng, Yunze Xiao, Aosong Feng, Dairui Liu, Yun Xing, Junjue Wang, Fan Gao, et al. MMLU-ProX: A Multilingual Benchmark for Advanced Large Language Model Evaluation. *arXiv preprint arXiv:2503.10497*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling Data for Software Engineering Agents. *arXiv preprint arXiv:2504.21798*, 2025b.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Your Efficient RL Framework Secretly Brings You Off-Policy RL Training, August 2025. URL https://fengyao.notion.site/off-policy-rl.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. *arXiv preprint arXiv:2309.12284*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Timur Zaharov, Konstantin Korolev, and Aleksandr Nikolich. Physics Big, 2024. URL https://huggingface.co/datasets/Vikhrmodels/physics_big.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Yonghao Li, Zhuohan Chen, Zhewei Wong, Siyuan Zhuang, Yakun Shao, Kai Xu, Zhenyu Zhang, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2309.11998*, 2023.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*, 2023.

A. Base model evaluations

For completeness, Table 5 presents the evaluation results for the base model checkpoint used to initialize the alignment process (referred to as the pre-alignment base). During development, we identified limitations in this model's performance on few key benchmarks, which motivated the training of the improved base model intended for release (as evaluated in Table 2).

Unlike the pre-alignment base, which trailed Qwen3, the improved checkpoint surpasses Qwen3 in the average performance on Code and General Knowledge tasks. The lead in Math tasks has also significantly widened compared to the pre-alignment base. In Multilingual benchmarks, while Qwen3 retains a lead on the MMLU Global Lite task, the improved checkpoint has surpassed Qwen3 on the MGSM task. The only significant regression is in Long Context, where the improved checkpoint shows a slight performance drop compared to the pre-alignment base, though it still maintains a commanding margin over Qwen3.

Task	Qwen3	N-3-Nano-Pre-Align
General Knowledge		
MMLU (5-shot, acc)	**81.07**	78.44
MMLU-Pro (5-shot, CoT EM)	**61.71**	61.39
AGIEval-En (3/5-shot, CoT acc)	63.12	**65.62**
Code		
HumanEval (0-shot)	**70.73**	69.51
MBPP-Sanitized (3-shot)	**73.15**	71.21
Math		
GSM8K (8-shot, acc)	**89.01**	87.04
MATH (4-shot, acc)	61.14	**80.80**
MATH-500 (4-shot, avg@32)	55.08	**72.79**
Commonsense Understanding		
ARC-Challenge (25-shot, acc_norm)	**94.45**	91.81
HellaSwag (10-shot, acc_norm)	83.14	**86.08**
OpenBookQA (0-shot, acc_norm)	44.80	**46.60**
PIQA (0-shot, acc_norm)	81.01	**83.68**
WinoGrande (5-shot, acc)	78.22	**79.08**
Reading Comprehension		
RACE (0-shot, acc)	**90.05**	87.56
Multilingual		
MMLU Global Lite (5-shot, avg acc)	**76.84**	75.69
MGSM (8-shot, avg acc)	**82.53**	78.93
Long Context		
RULER (64K, 0-shot, acc)	63.55	**88.94**
RULER (128K, 0-shot, acc)	60.69	**86.78**
RULER (256K, 0-shot, acc)	-	**79.15**

Table 5 | Comparison of **Qwen3-30B-A3B-Base** and the **Nemotron 3 Nano** pre-alignment base checkpoint (the specific checkpoint used to initialize the alignment pipeline). Best results between these two are marked in bold.

B. MMLU-redux evaluation

We developed the following two variants of MMLU-redux:

(1) MMLU-redux CoT. We created this variant due to the observation that many STEM questions intrinsically require step-by-step reasoning for successful resolution, which is not adequately captured by the original multiple-choice, no chain-of-thought format. The model might arrive at some answers through guessing or memorization. Therefore, we created five exemplars per subject, each accompanied by a detailed step-by-step solution. This allows us to evaluate models using a 5-shot chain-of-thought setting.

(2) MMLU-redux Tweak. As MMLU's widespread use increases the risk of overfitting and benchmark saturation from extensive tuning, we introduced this variant to more rigorously evaluate model performance on similar yet new examples that closely match the original in difficulty, style, structure, and format. We modified the original test examples using Qwen3-235B-A22B-Thinking-2507 to assess the same underlying concepts, ideas, and skills while altering specific details such as numerical values and equations.

The evaluation results are presented in Table 6. Overall, enabling CoT reasoning yields a substantial accuracy boost, especially on STEM subjects. Our model demonstrates a larger gain from CoT compared to Qwen (an average improvement of +5.27 versus +0.79, respectively). In addition, we observe a significant increase in the Professional Accounting task under the Other category, with an improvement from 64.00 to 77.00 (+13.00), as this task also relies heavily on calculation skills.

On MMLU-redux Tweak, both models achieve noticeable gains across non-STEM categories, likely because many non-STEM questions assess domain knowledge, and the tweaked questions were generated using Qwen3-235B-A22B-Thinking-2507, whose knowledge may align more closely with the evaluated models. We observe a divergent trend on STEM: Qwen's accuracy decreases marginally (−0.83), while our model's score increases by 5.31.

	MMLU-redux		**MMLU-redux CoT**		**MMLU-redux Tweak**	
	Qwen	Ours	Qwen	Ours	Qwen	Ours
STEM	81.05	74.42	84.05 (+3.00)	87.26 (+12.84)	80.22 (−0.83)	79.26 (+4.84)
Humanities	82.31	80.46	83.16 (+0.85)	81.23 (+0.77)	85.04 (+2.73)	84.04 (+3.58)
Social Sciences	86.83	84.42	85.92 (−0.91)	85.50 (+1.08)	89.36 (+2.53)	89.70 (+5.28)
Other	80.23	77.85	79.85 (−0.38)	80.38 (+2.53)	82.43 (+2.20)	84.00 (+6.15)
All	82.37	78.68	83.16 (+0.79)	83.95 (+5.27)	83.76 (+1.39)	83.64 (+4.96)

Table 6 | Evaluation results on MMLU-redux and two variants. "Qwen" refers to the Qwen3-30B-A3B-Base model. "Ours" denotes our base model checkpoint used in the ablation study, which was trained on a data blend that differs slightly from the one used for our final model as the ablation study was conducted alongside training.

C. DPO for Reducing Tool Hallucination

Reducing hallucinated tool usage is one of the key objectives of our alignment experiments. Although our released model does not rely on DPO, because reinforcement learning (RL) already achieved comparable performance, we nevertheless explored DPO as an additional technique due to its simplicity and minimal computational overhead. As shown later, even a very small amount of DPO training yields meaningful reductions in hallucinated tool calls and improves reasoning stability. To support this analysis, we first define what constitutes hallucinated tool usage in our evaluation.

Definition of Tool Hallucination and Hallucination Rate. We define **tool hallucination** as any instance in which the model attempts to invoke a tool despite no tools being declared in the system message. Under the *No-Tools* and *Hallucination-Penalty* settings, the model is expected to rely entirely on internal reasoning; therefore, any output containing a tool call, such as a Python execution request, a search invocation, or any tool-specific API format, is treated as a hallucination.

The **tool hallucination rate** is the proportion of evaluation samples in which such unintended tool calls occur. A higher rate indicates inappropriate tool triggering, whereas a near-zero rate reflects strong calibration and reliable adherence to environment constraints.

DPO Data Construction. To study how DPO affects tool-use calibration and reasoning performance, we constructed a DPO dataset using 2,000 reasoning tasks: 1,000 mathematics problems and 1,000 STEM multi-choice questions. For each problem, the model generated 32 on-policy solutions, providing a diverse set of candidate behaviors. These raw generations were then processed through our DPO data-construction pipeline, assigning preference labels according to correctness and tool-usage conditions, which produced approximately 50k preference samples in total. We later found that the model's improvements persisted even when using substantially smaller datasets; in fact, training with as few as 10k preference samples (or even fewer) yielded similar benefits. This further underscores the low computational cost and high sample efficiency of DPO in our setting. To study tool-use alignment, we organized the data into three categories: (1) No-Tools, where the system message does not expose tools and correctness alone determines preference labels; (2) With-Tools, where tools are available and labels depend only on the correctness of the final answer; and (3) Hallucination-Penalty, where tools are not declared and any hallucinated tool invocation is labeled as a negative preference. This structure allows us to jointly evaluate pure reasoning ability, tool-assisted reasoning, and calibration of tool usage, while providing a rich set of preference signals derived from diverse on-policy model behaviors.

Training Setup. For our DPO experiments, we used a lightweight training configuration designed to minimally perturb the model after SFT while still providing a meaningful preference-learning signal. Specifically, we trained with a learning rate of 3e-6, a batch size of 128, and 50 training steps. We set the SFT loss coefficient to 0.2, the preference (DPO) loss coefficient to 1.0, and the KL loss coefficient to 0.05. This setup emphasizes preference learning while retaining a small supervised loss to stabilize outputs and a modest KL penalty to prevent excessive deviation from the base model. This configuration emphasizes preference learning while retaining a small supervised loss to stabilize outputs and a modest KL penalty to prevent excessive deviation from the base model.

Results. Table 7 shows the impact of applying a small amount of DPO training on both reasoning accuracy and hallucinated tool usage. Despite using only 50 training steps with a modest learning rate, we observe consistent improvements across all evaluated benchmarks.

For AIME25, accuracy increases from 80.88% to 84.58%, indicating that DPO not only suppresses undesirable tool-related behaviors but also enhances overall solution quality. Notably, the hallucination rate, which is already low in this setting, is reduced from 1.25% to 0%, fully eliminating spurious tool invocation.

On GPQA, which is more challenging and shows higher baseline hallucination, DPO again yields substantial gains. Accuracy improves from 65.15% to 69.19%, and the hallucination rate drops dramatically from 8.33% to just 0.7%. This confirms that preference-based fine-tuning is particularly effective in settings where the model is prone to uncertainty or over-triggering tool calls.

Overall, the results demonstrate that even minimal DPO training can meaningfully reduce hallucinated tool usage while simultaneously improving reasoning accuracy. This suggests that DPO provides a valuable complementary signal to RL-based alignment, strengthening both model reliability

and calibration with negligible computational cost.

	Accuracy		Hallucination Rate	
	Before DPO	After DPO	Before DPO	After DPO
AIME25 (no tools)	80.88	84.58	1.25%	0%
GPQA (no tools)	65.15	69.19	8.33%	0.7%

Table 7 | Evaluation results on DPO experiments.

D. Safety Preference Data

For the RLHF stage, reward model training data comprises of the same underlying datasets used in the SFT safety subset, leading to a similar distribution for the starting seed prompts. Response generation is more nuanced, to handle over-refusals and harmful engagements as the rejected responses.

- For **harmful prompts**, chosen responses are generated with a similar strategy as the SFT responses. The rejected responses are unsafe model outputs, generated via two methods: (i) applying jailbreak templates to produce harmful completions, and (ii) directly prompting the model and using a content safety moderation classifier to detect cases of harmful outputs.
- For **safe prompts**, chosen responses are generated by passing the safe prompt as-is to the underlying model, and using a content safety moderation classifier to ensure safe responses. The rejected responses are generated by applying refusal prompt templates, resulting in over-refusals.

The resulting response pairs are thus annotated using a preference-based scheme: for harmful prompts, <safe, unsafe> completions are labeled as the <chosen, rejected> pairs. For safe prompts, <safe, over-refusal> completions are annotated similarly as <chosen, rejected> pairs. This approach supports training reward models for both robust safety alignment and mitigating over-refusal behaviors.

To ensure diversity, we generate the chosen and rejected response pairs for each prompt using five (5) different open-source models, followed by applying necessary filters to keep only safe (for chosen) and unsafe or over-refusal responses (for rejected) to build a list of candidate chosen and rejected responses. Finally, one chosen and rejected response pair per prompt is chosen randomly from the candidates.

E. Prompt Sensitivity Analysis

Benchmark	**N-3-Nano**	**Qwen3**	**GPT-OSS**
GPQA (no tools)	0.42	0.59	1.91
MMLU-Pro	0.41	0.31	1.46
Comp-Math-24-25 (no tools)	0.77	0.51	1.14
LiveCodeBench (v6 2024-08↔2025-05)	0.83	1.05	1.02

Table 8 | Prompt sensitivity for Nemotron 3 Nano, Qwen3-30B-A3B-Thinking-2507 and GPT-OSS 20B (lower is better). (Comp-Math-24-25 contains AIME24, AIME25, HMMT 2024 Feb., Nov. and 2025 Feb. datasets).

LLM predictions can be sensitive to minor changes to the input (Nalbandyan et al., 2025). Even simple, non-adversarial edits (e.g., changes in prompt wording, answer formatting instructions, or problem placement relative to the prompt) can shift the model's outputs enough to change individual predictions and, in aggregate, benchmark accuracy. To reduce the risk of over- or under-estimating accuracy due to a single prompt choice, we evaluate models using multiple prompts. This better reflects model stability under routine, realistic prompt variations.

To measure prompt sensitivity, we construct a set of prompts for each dataset varying in wording, instruction granularity (minimal vs. detailed), problem placement (before, middle, or after the prompt), and answer formatting. For each prompt, we compute mean accuracy across eight seeds, and we use the standard deviation of prompt averages as the prompt sensitivity metric. Prompt sensitivity results are presented in Table 8. With sensitivity scores below 1 across all datasets, Nemotron 3 Nano shows strong stability and robustness to changes in the prompt.