NEMOTRON-CASCADE 2: POST-TRAINING LLMS WITH CASCADE RL AND MULTI-DOMAIN ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Nemotron-Cascade 2, an open 30B MoE model with 3B activated parameters that delivers best-in-class reasoning and strong agentic capabilities. Despite its compact size, its mathematical and coding reasoning performance approaches that of frontier open models. It is the second open-weight LLM, after DeepSeek-V3.2-Speciale-671B-A37B, to achieve Gold Medal-level performance in the 2025 International Mathematical Olympiad (IMO), the International Olympiad in Informatics (IOI), and the ICPC World Finals, demonstrating remarkably high intelligence density with 20× fewer parameters. In contrast to Nemotron-Cascade 1 (Wang et al., 2026), the key technical advancements are as follows. After SFT on a more meticulously curated dataset, we substantially scale Cascade RL to cover a broader spectrum of reasoning and agentic domains, grouping compatible domains with similar response lengths and verification costs. Furthermore, we incorporate multi-domain on-policy distillation from the strongest intermediate checkpoints in each domain to efficiently recover from potential benchmark regressions and sustain performance gains throughout the Cascade RL process. We release model checkpoint and training data for research community.

1 INTRODUCTION

Reinforcement Learning (RL) (Ouyang et al., 2022; Guo et al., 2025) has become central to LLM post-training, driving recent progress in reasoning, coding, and agentic capabilities. Scaling RL across a broad set of domains, however, remains challenging: different tasks often require different reward signals, response lengths, verifier costs, and environment dynamics. Jointly training all domains can introduce cross-domain interference, while training each domain independently or strictly sequentially can become prohibitively expensive as the number of environments grows.

Wang et al. (2026) introduced Cascade RL, a framework that orchestrates sequential, domain-wise RL training across specialized task domains. Cascade RL simplifies the engineering complexity associated with mixed-domain RL, while achieving state-of-the-art performance across a wide range of benchmarks. The advantages of Cascade RL are threefold. First, domain-specific RL stages are remarkably resistant to *catastrophic forgetting*. They rarely degrade benchmark performance attained in earlier domains and may even improve it. Second, it allows RL hyperparameters and the training curriculum to be carefully tailored to each specific domain, enabling optimized learning dynamics and improved final performance. Third, task homogeneity within each RL stage also yields substantial compute savings, as response lengths and verification wall-clock times are more uniform within a domain than across multiple domains trained jointly. Despite these advantages, scaling Cascade RL to many more domains and training environments introduces new challenges. A strictly sequential pipeline increases training latency as more domains are added, and naively combining domains within a stage can reintroduce interference between heterogeneous reward signals and compromise the training curriculum for each domain.

In this work, we introduce Nemotron-Cascade 2, an open 30B Mixture-of-Experts (MoE) model with 3B activated parameters that scales Cascade RL to a broader and more diverse set of domains. Our training pipeline extends Cascade RL along two axes, as illustrated in Figure 1. First, to avoid the full cost of assigning every environment to its own sequential stage, we introduce in-stage multi-domain

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

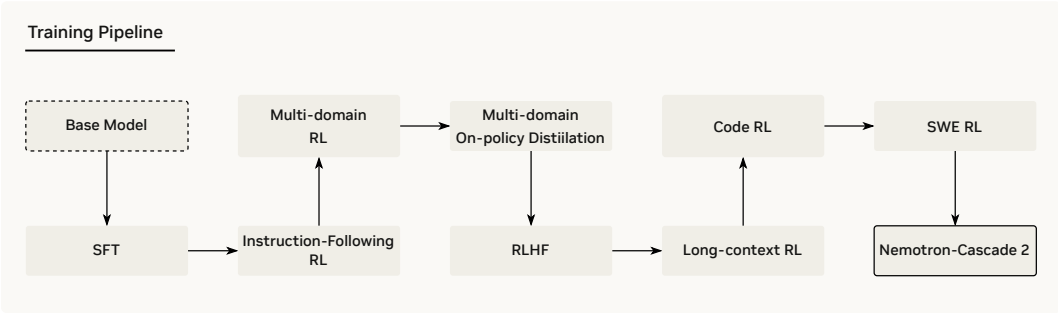


Figure 1: Nemotron-Cascade 2 training pipeline. Compared with prior Cascade RL, our pipeline introduces multi-domain RL to jointly optimize compatible environments within a stage, improving coverage and efficiency while preserving much of the stability of domain-specialized training. We further insert Multi-domain On-Policy Distillation (MOPD) as a stabilization stage to distill the strongest intermediate domain teachers into one policy, recovering reasoning regressions caused by response-length compression.

RL. Instead of training each domain separately, we jointly train groups of domains with similar response lengths and comparable verification costs. This allows compatible environments to be optimized in parallel, improving coverage and training efficiency while retaining much of the stability and tunability of Cascade RL.

However, when scaling Cascade RL with in-stage multi-domain training, we find that task mixture and training hyperparameters can substantially shape the model's response-length distribution through the aggregation of policy gradients. In particular, the multi-domain stage combines tasks whose reward structures tend to favor shorter responses, causing the policy to converge to an average response length of roughly 32K tokens. While this length distribution is sufficient for many domains, it is suboptimal for competitive mathematical reasoning, where the strongest performance often relies on much longer reasoning trajectories (e.g., 131K). As a result, this multi-domain stage alone regresses AIME accuracy from 92 to 88. To address this issue, we insert a single Multi-domain On-Policy Distillation (MOPD) stage (Xiao et al., 2026; Zeng et al., 2026) immediately after multi-domain RL. MOPD distills the strongest per-domain intermediate teachers back into the policy, recovering the lost math reasoning performance and providing a stronger initialization for the subsequent RLHF stage.

The resulting model, Nemotron-Cascade 2-30B-A3B, achieves gold-medal performance on both IMO 2025 and IOI 2025. To our knowledge, it is the second open-weight LLM to reach this level after DeepSeek-V3.2-Speciale-671B-A37B (Liu et al., 2025), while using roughly $20\times$ fewer total parameters and $12\times$ fewer activated parameters. Beyond math and code, Nemotron-Cascade 2 also delivers strong performance across alignment, instruction following, long-context tasks up to 1M tokens, and agentic benchmarks; full results are reported in **Table 2**. To enable the community to reproduce, analyze, and extend the proposed pipeline, we open-sourced the model and training data.

We organize the remainder of this paper as follows. §2 covers supervised fine-tuning. §3 presents Cascade RL integrated with MOPD. §4 presents the results on comprehensive benchmarks, including IMO, IOI and ICPC World Finals results. We conclude the paper in §5.

2 Supervised Fine-Tuning

Supervised fine-tuning (SFT) is the first stage of our post-training pipeline and provides the broad behavioral initialization for the subsequent Cascade RL and multi-domain on-policy distillation stages. Rather than optimizing for a single benchmark family, this stage prepares the model for the diverse interaction modes required later in the pipeline, including long-form reasoning, instruction following, tool use, long-context processing, software engineering, and terminal-based agentic tasks.

Compared with Nemotron-Cascade 1 (Wang et al., 2026), the Cascade-2 SFT recipe expands both the reasoning and agentic data mixtures. The reasoning mixture includes tool-use and non-tool math traces, natural-language proof generation and verification, competitive-programming traces in Python and C++14, scientific coding, and science QA. The general and agentic mixture includes multi-turn dialogue, reasoning-on and reasoning-off instruction following, safety data, conversational tool use, software-engineering trajectories from both agentic and agentless scaffolds, and terminal-agent tasks.

We also simplify the chat template and standardize tool calling so that the same model can support thinking and non-thinking modes as well as structured tool invocation. In particular, the model uses explicit thinking markers to distinguish reasoning modes, while available tools are specified in the system prompt and invoked through structured tool-call tags. Detailed data construction, filtering procedures, chat-template changes are provided in Appendix B.

We train a single SFT stage on a broad data mixture of over 26M training instances per epoch. Samples are packed into sequences of up to 256K tokens, enabling the model to learn from both standard instruction-response examples and long-context or agentic trajectories. The best checkpoint is selected after approximately 1.5 epochs. SFT training is performed on NVIDIA H100 GPUs, with full training hyperparameters provided in Appendix H.

3 Cascade RL and Multi-Domain On-Policy Distillation

After supervised fine-tuning, we apply Cascade RL to specialize the model through a sequence of domain-wise RL stages rather than a single heterogeneous RL mixture. As illustrated in Figure 1, the Cascade-2 pipeline proceeds through instruction-following RL, multi-domain RL, multi-domain on-policy distillation (MOPD), RLHF, long-context RL, code RL, and SWE RL. Compared with Nemotron-Cascade 1 (Wang et al., 2026), Cascade-2 extends the framework with in-stage multi-domain RL for efficient scaling and MOPD as a stabilization stage for recovering from capability regressions. The full RL pipeline uses about 13K H100 GPU hours on 128 GPUs (per-stage costs in Appendix C.9). Detailed dataset curation and training recipes are provided in Appendix C, and ablations of MOPD are presented in §3.5.

3.1 What determines the ordering of Cascade RL

The optimal stage ordering in Cascade RL is not fixed; it depends on the model's behavior and learning dynamics. Compared to Wang et al. (2026), Nemotron-Cascade 2 uses higher-quality SFT data and a substantially larger set of RL environments, which motivates a revised ordering.

Mitigating inter-domain interference. Our ordering is primarily guided by the need to reduce catastrophic forgetting and negative transfer across increasingly diverse RL domains. Cascade RL enables fine-grained analysis of such domain interactions, including the tension between strict instruction following in IF-RL and human preference alignment in RLHF. We further observe that placing IF-RL and RLHF after the long-reasoning stages substantially shortens reasoning traces and degrades reasoning benchmarks. We therefore schedule both stages before code RL. This establishes foundational behaviors, namely concise and accurate responses, before the specialized reasoning stages, which can then extend the reasoning length without unnecessarily degrading other domains.

Scaling via multi-domain integration. When domains are empirically non-conflicting or complementary, Cascade RL can merge them into a multi-domain RL stage, which improves scalability as environments grow while maintaining broad benchmark performance (§3.4). Concretely, we merge domains with similar expected output lengths and reasoning demands: MCQA, tool calling, and structured output all elicit short answers and share a 49K maximum length, whereas code and SWE RL require considerably longer reasoning (over 100K tokens) and are trained in separate stages.

Stabilizing through on-policy distillation. We further show that MOPD (§3.5) provides an important stabilization point in the cascade. It recovers benchmark performance that may regress during earlier specialized stages, yielding a more balanced and robust final policy.

3.2 On-Policy RL Objective

Across RL stages, we use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with strict on-policy sampling. At each iteration, the current policy π_θ samples a group of G rollouts for each prompt, and the model is updated once using those same rollouts. This keeps the data-collection policy and optimization policy matched, improving stability across the heterogeneous stages.

For a prompt q, sampled responses $\{o_i\}_{i=1}^{G}$, and scalar rewards $\{r_i\}_{i=1}^{G}$, we use group-normalized advantages

$$\hat{A}_i = \frac{r_i - \mathrm{mean}(\{r_j\}_{j=1}^{G})}{\mathrm{std}(\{r_j\}_{j=1}^{G})}. \tag{1}$$

Table 1: Stage-wise benchmark performance throughout the training pipeline.

Stage	AIME25 avg@64	LCB v5 avg@8	ArenaHard v2 Hard / Creative	IFBench Prompt / Instruction	AA-LCR avg@16	τ^2-Bench Overall	SWE-bench avg@4
SFT CKPT	92.9	83.8	78.4 / 65.7	56.0 / 61.9	26.4	51.3	45.4
IF-RL + Multi-domain RL	89.4	88.2	70.4 / 39.7	84.4 / 86.0	32.3	57.1	–
MOPD	92.3	88.2	85.5 / 71.0	81.7 / 83.8	33.8	57.0	46.7
RLHF	92.5	86.3	87.5 / 80.3	80.0 / 82.2	31.7	56.2	48.1
Long Context RL	91.9	86.9	87.3 / 78.3	81.3 / 83.1	35.1	58.8	48.9
Code RL	92.4	88.7	88.0 / 81.5	81.8 / 83.8	38.5	59.5	48.9
SWE RL	92.4	88.4	88.2 / 78.7	82.9 / 84.7	39.1	58.9	50.2

The token-level GRPO objective is

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\ \{o_i\}_{i=1}^{G}\sim\pi_\theta(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \hat{A}_i \log \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \right]. \tag{2}$$

For RLVR stages, r_i is provided by verifiable checkers or environment feedback; for RLHF, r_i is produced by a generative reward model. This shared on-policy objective provides the optimization backbone for all subsequent stages.

3.3 Instruction-Following RL

We begin with verifiable instruction-following RL (IF-RL), which trains the model to satisfy explicit constraints such as formatting requirements, length limits, and structured response conditions. We place IF-RL first because strict instruction adherence can conflict with later human-preference optimization, while subsequent RLHF can recover preference quality with little degradation to instruction-following scores.

IF-RL uses automatically verifiable instruction-following prompts from (Blakeman et al., 2025a), dynamic filtering (Yu et al., 2025) to retain prompts with informative rollout variation, and an overlong penalty to discourage unnecessary response growth. This stage improves IFBench (Pyatkin et al., 2025) from 56% to 83.1%. The key role of IF-RL is to establish a strong instruction-following foundation before the model enters broader multi-domain and preference-optimization stages.

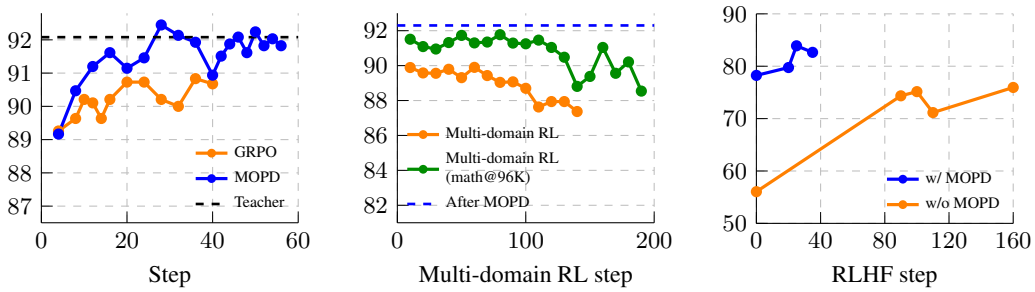
3.4 Multi-Domain RL

A strictly sequential Cascade RL pipeline becomes increasingly expensive as the number of training environments grows. To scale Cascade RL more efficiently, we introduce an in-stage multi-domain RL phase that jointly trains compatible verifiable domains. Rather than mixing all tasks indiscriminately, we group domains only when they have similar response formats, response lengths, and verification costs, and when empirical training does not reveal strong cross-domain interference.

This stage jointly trains STEM multiple-choice question answering, agentic tool calling in the Workplace Assistant setup, and structured-output instruction following, with a mixture of approximately 55% MCQA, 30% agentic tool calling, and 15% structured-output data drawn from Blakeman et al. (2025a). We found improvements on MMLU-Pro, τ^2-Bench, and IFBench, showing that Cascade RL can scale through multi-domain stages when the grouped domains are compatible, meanwhile do not degrade performance on other benchmarks. However, there is an exception for math benchmarks, as this stage naturally favors shorter responses under its task mixture and hyperparameters, which motivates the MOPD stage described next.

3.5 Multi-Domain On-Policy Distillation

While Cascade RL substantially reduces catastrophic forgetting compared with sequential RL in an arbitrary order, it does not fully eliminate capability drift as the number of training environments grows, and the dominant trade-off differs by stage. RLVR stages tend to reduce entropy and shorten reasoning traces, which hurts competitive math, whereas RLHF can trade off against strict instruction following. In our pipeline, IF-RL and multi-domain RL lower AIME25 from 92.9 to 89.4 and ArenaHard v2 Creative Writing from 65.7 to 39.7 (Table 1). We thus insert a multi-domain on-policy



(a) MOPD vs. GRPO (AIME25) (b) Math length in multi-domain RL (AIME25) (c) RLHF w/ vs. w/o MOPD (ArenaHard v2)

Figure 2: MOPD analysis. (a) Under math-only training, MOPD reaches teacher-level AIME25 in fewer steps than GRPO. (b) AIME25 during multi-domain RL. *Multi-domain RL*: the default multi-domain RL stage. *Multi-domain RL (math@96K)*: the same stage with math prompts at 3.7% of the mixture and a 96K maximum response length for math (49K for the other domains). The longer math budget delays but does not prevent the regression; the dashed line marks AIME25 after MOPD (92.3). (c) RLHF from the MOPD checkpoint surpasses RLHF without MOPD with $6\times$ fewer steps. ArenaHard v2 is the average of Hard Prompt and Creative Writing.

distillation (MOPD) stage (Xiao et al., 2026; Yang et al., 2025; Agarwal et al., 2024; Gu et al., 2024; Lu & Lab, 2025; Zeng et al., 2026) to rebalance capabilities before the remaining RL stages.

MOPD fits naturally into Cascade RL. Teachers are taken directly from the pipeline as the strongest checkpoint for each capability domain, which removes the need for external model families; moreover, because all teachers share the student's SFT initialization and tokenizer, distillation introduces little distribution shift. We use three teachers: the SFT checkpoint for math, which already exhibits strong mathematical reasoning; an RLHF checkpoint trained from SFT; and the best IF-RL + multi-domain RL checkpoint. Each prompt is routed to one teacher by a predefined rule based on its source data pool. In preliminary experiments, distilling from the math teacher alone degraded IFBench, motivating multiple teachers. For each prompt x, the student samples a response $y = (y_1, \ldots, y_T)$ on-policy, and the teacher π^{domain} assigned to the prompt's domain provides a token-level advantage

$$a_t^{\text{MOPD}} = \log \pi^{\text{domain}}(y_t \mid x, y_{<t}) - \log \pi^{\text{train}}(y_t \mid x, y_{<t}), \tag{3}$$

which we optimize with a policy-gradient surrogate. Unlike outcome rewards that assign one scalar to an entire response, this gives a dense learning signal on every student-sampled token. Details are given in Appendix C.4.

MOPD recovers regressions in far fewer steps. Because every token receives a learning signal, MOPD converges much faster than outcome-reward RL. Under math-only training from the same checkpoint, MOPD reaches the teacher's AIME25 score within 30 steps, whereas GRPO plateaus below 91 (Figure 2(a)). On ArenaHard v2, MOPD reaches 85.5 on Hard Prompt and 71.0 on Creative Writing in 52 steps, whereas RLHF from the same checkpoint reaches only 80.7 and 71.2 after 160 steps (Table 6). In the full pipeline, MOPD restores AIME25 to 92.3 while also improving ArenaHard v2 (Table 1).

Extending the math length budget in multi-domain RL is not a substitute. Since the regression coincides with shorter responses, a natural alternative to an extra stage is a length-aware mixture: we reduce the share of math prompts to 3.7% of the mixture and raise their maximum response length to 96K tokens, while keeping 49K for the other domains. This delays but does not prevent the regression (Figure 2(b)). With the 96K math budget, AIME25 holds around 91–92 for the first 120 steps, but starts to drop after about 130 steps and reaches 88.5 by step 190. This suggests that the regression is not solely attributable to the length budget: because the other domains rarely require deep reasoning, the shared policy update shifts the model away from long, structured reasoning, even when math rollouts are permitted to be long. MOPD circumvents this issue by distilling from a teacher that exhibits the desired long-reasoning behavior.

MOPD provides a stronger initialization for RLHF. Applying RLHF directly after multi-domain RL, without MOPD, converges more slowly and to a lower level (Figure 2(c)): its ArenaHard v2

average plateaus at 76.0 after 160 steps, whereas RLHF from the MOPD checkpoint reaches 83.9 in 25 steps. Combining the two stages gives the best result on both subsets (Table 1).

3.6 RLHF

After MOPD, we apply RLHF to improve human-preference alignment, creative writing, and non-verifiable reasoning behavior. The training prompts for this stage are drawn from Blakeman et al. (2025a). This stage uses a generative reward model (GenRM) powered by Qwen3-235B-A22B (Qwen-Team, 2025) and trained under the HelpSteer3 framework (Wang et al., 2025c). Given a prompt and multiple candidate responses, GenRM evaluates their relative quality and provides preference rewards for GRPO training.

We score pairwise comparisons among rollouts, aggregate the resulting rewards, and apply length-normalized reward adjustment together with a quality-gated conciseness bonus to control unnecessary response growth. We train RLHF only in thinking mode: although mixing thinking and non-thinking modes improves some alignment metrics, we find that it degrades instruction following in ways that later stages do not fully recover. We use a maximum response length of 16K during rollout without overlong filtering. Training converges in around 30 steps, benefiting from the higher ArenaHard score achieved by the prior MOPD stage.

3.7 Long-Context RL

We next apply a dedicated long-context RL stage to improve reasoning over long input sequences. This stage uses long-context-only data from the NVIDIA Nano-v3 RL blend (Blakeman et al., 2025a), together with the NeMo-Gym environment (NVIDIA, 2025c) and an LLM judge (Qwen3-235B-A22B) for question-answering tasks. During training, input sequences are limited to 32K tokens and the maximum total sequence length is set to 49K tokens without applying overlong filtering.

We isolate long-context RL as its own stage because mixing long-context data with unrelated domains causes regressions on non-long-context benchmarks. We train this stage for 30 steps as we observe a rapid increase in generated tokens beyond this point.

3.8 Code RL

Code RL specializes the model for challenging competitive programming. We construct a compact training set from the Nemotron-Cascade coding corpus (Wang et al., 2026), filtering out prompts that GPT-OSS-120B (Agarwal et al., 2025) solves correctly in all 8 out of 8 rollouts. This yields approximately 3.5K difficult prompts with robust test cases, providing sparse but high-quality verifiable rewards.

We use a strict binary reward based on test-case correctness to avoid potential reward hacking. We increase the maximum response length to 118K tokens, and sample 16 rollouts per prompt to enable the policy to better capture sparse reward signals on extremely difficult problems. An asynchronous verifier supports the resulting execution load of 2,048 code executions per RL step, which completes each batch in 427.2 seconds across 384 CPU cores.

3.9 SWE RL

Finally, we apply SWE RL to adapt the model to repository-level software engineering tasks. In the main Cascade-2 pipeline, we use execution-based agentic RL with an OpenHands-style scaffold (Wang et al., 2025a), where the agent inspects files, edits code, and runs tests over multiple turns. Compilation results and unit-test outcomes provide verifiable rewards, directly aligning training with SWE-bench-style evaluation. The training data is drawn from SWE-Gym (Pan* et al., 2025) and R2E-Subset (Jain et al., 2025). To keep the training set challenging, we use an intermediate checkpoint to filter out samples with a 100% pass rate over 16 rollouts. We also randomly discard 90% of samples with a 0% pass rate. We train with 16 prompts and 64 rollouts per prompt, using a maximum context length of 256K. The agent is allowed up to 200 turns per rollout.

Table 2: **Main results**. Nemotron-Cascade-2-30B-A3B achieves gold-medal performance on both IMO 2025 and IOI 2025 while using only 30B total parameters and 3B activated parameters, demonstrating strong intelligence density. Numbers in parentheses indicate Tool-Integrated Reasoning (TIR) results. For baseline models, we report official numbers when available; otherwise, we evaluate them using recommended settings, marked by ‡.

Benchmark	Nemotron-3-Nano 30B-A3B	Nemotron-3-Super 120B-A12B	Qwen3.5 35B-A3B	Nemotron-Cascade-2 30B-A3B
Math				
IMO 2025	–	–	–	🥇 **35 pts**
IMO AnswerBench	70.4‡	77.2‡	74.8‡	79.3
IMO ProofBench	–	–	–	72.9
AIME 2025	89.1	90.2	91.9‡	92.4 (98.6)†
AIME 2026	89.9‡	89.8‡	91.1‡	90.9 (95.0)†
HMMT Feb25	84.6‡	93.7	89.0	94.6
Code Reasoning				
IOI 2025	–	–	348.6‡	🥇 **439.28**
ICPC World Finals 2025	–	–	–	🥇 **10/12**
LiveCodeBench v6 (2408-2505)	68.3	78.7	74.6	87.2 (88.4)†
LiveCodeBenchPro 25Q2 (Easy)	54.5‡	81.7‡	81.1‡	87.0 (89.3)†
LiveCodeBenchPro 25Q2 (Med)	3.50‡	23.2‡	17.8‡	27.6 (36.8)†
SciCode	33.3	42.1	38.0	36.4
Knowledge & STEM				
MMLU-Redux	–	–	93.3	86.3
MMLU-Pro	78.3	83.7	85.3	79.8
GPQA-Diamond	73.0	79.2	84.2	76.1
HLE (no tool)	10.6	18.3	22.4	17.7
Alignment & Instruction Following				
ArenaHard v2 (Avg.)	67.7	–	65.4‡	83.5
– Hard Prompt	72.1	73.9	64.5‡	88.2
– Creative Writing	63.2	–	66.3‡	78.7
IFBench (prompt)	71.5	72.6	70.2	82.9
Scale AI Multi-Challenge	38.5	55.2	60.0	45.3
Long Context & Context Learning				
AA-LCR	35.9	58.3	58.5	39.1
LongBench v2	39.6	–	59.0	40.3
NIAH@1M (RULER Subset)	94.8	98.3	94.3‡	99.0
CL-Bench	12.0‡	–	15.5‡	12.2
Agentic				
BFCL v4	53.8	–	67.3	52.9
τ^2-Bench	49.0	61.2	81.2	58.9
Terminal Bench 2.0	8.5	31.0	40.5	21.1
SWE Verified (OpenHands)	38.8	60.5	69.2	50.2
Multilingual				
MMLU-ProX	59.5	79.4	81.0	72.5
WMT24++ (en -> xx)	86.2	86.7	87.6‡	84.1

We also study agentless code-repair RL as an auxiliary analysis. Although it is not part of the main pipeline, we find that it improves both agentless and OpenHands-style agentic evaluation, suggesting that stronger repository-level repair skills transfer across scaffolds.

4 Results

We evaluate Nemotron-Cascade-2-30B-A3B on a comprehensive suite of benchmarks covering math and coding reasoning, knowledge and STEM, alignment and instruction following, long-context understanding and in-context learning, multilingual capabilities, and agentic tasks. The main results are shown in Table 2, with benchmark details and evaluation setups provided in Appendix F.

Table 3: Performance of Nemotron-Cascade-2-30B-A3B model on IMO 2025, IOI 2025, and ICPC World Finals 2025 competitions. Nemotron-Cascade-2 model achieved solid gold medal on all these top-tier competitions. Our IMO 2025 solutions are evaluated by human expert (IMO [20**] Gold medalist) while IOI 2025 and ICPCWF 2025 solutions are verified through OnlineJudge with official testcases. *Single pass* reports the score of one sample per problem without test-time scaling.

Competition	P1	P2	P3	P4	P5	P6	Overall	Medal	Single pass
IMO 2025	7	7†	7	7	7	0	35/42	Gold	5/42
IOI 2025	39	88.53	100	100	28.75	83	439.28/600	Gold	262/600

Competition	A	B	C	D	E	F	G	H	I	J	K	L	**Overall**	**Medal**	**Single pass**
ICPC World Final 2025	+	–	+	+	+	+	–	+	+	+	+	+	10/12	Gold	7/12

† For IMO 2025 P2, we use an LLM grader with the reference solution and marking scheme from ProofBench (Ma et al., 2025). The model's solution follows an extensive analytic-geometry approach, making complete manual verification of all intermediate derivation steps difficult.

The most notable result is the model's performance on competition-level reasoning tasks. Despite using only 30B total parameters and 3B activated parameters, Nemotron-Cascade-2-30B-A3B achieves gold-medal-level performance on IMO 2025, IOI 2025, and ICPC World Finals 2025. These results place Nemotron-Cascade-2 among the strongest open-weight reasoning models, reaching a level previously demonstrated primarily by frontier proprietary systems (Gemini Team, 2025b) and much larger open-weight models (Liu et al., 2025). Detailed competition results are reported in Table 3. These results rely on substantial test-time scaling; without it, the single-pass scores are 5/42 on IMO, 262/600 on IOI, and 7/12 on ICPC.

Beyond these competitions, Table 2 shows that Cascade-2 achieves leading or highly competitive performance across mathematical reasoning, code reasoning, general alignment, and instruction following. Tool-integrated reasoning further improves several math and coding benchmarks.

The comparison with Nemotron-3-Nano-30B-A3B (Blakeman et al., 2025a) is particularly informative because both models are post-trained from the same pretrained base model, Nemotron-3-Nano-30B-A3B-Base (NVIDIA, 2025b). Cascade-2 improves over Nemotron-3-Nano on nearly all benchmarks, suggesting that the gains come primarily from the proposed Cascade RL plus MOPD training pipeline rather than from differences in pretraining.

At the same time, the results highlight several limitations. Cascade-2 remains behind Qwen3.5-35B-A3B and Nemotron-3-Super-120B-A12B on several knowledge-intensive, multilingual, long-context, and agentic benchmarks. These gaps suggest that stronger knowledge-intensive mid-training, multilingual training, and more targeted agentic RL are promising directions for future work.

4.1 International Mathematical Olympiad (IMO)

In Table 3, we evaluate Nemotron-Cascade-2-30B-A3B on the IMO 2025 problem set using a self-improving test-time scaling framework (Shao et al., 2025), where the model iteratively generates, verifies, and refines candidate solutions using its own feedback. Despite its relatively compact 30B-A3B scale, Nemotron-Cascade-2-30B-A3B solves the first five IMO 2025 problems, suggesting that effective inference-time scaling can substantially amplify the reasoning capability of smaller models. This trend is further reflected on IMO-ProofBench (Luong et al., 2025): with generate-verify-refine scaling, the model achieves 72.9, within 8 points of DeepSeek-Math-V2-671B-A37B while using 10$\times$ fewer active parameters, and surpasses QED-Nano-4B (LM-Provers et al., 2026) by 18 points overall. As shown in Figure 3, increasing test-time compute improves performance on IMO-ProofBench (Advanced) from 40.7 at round 1 to 53.4 at round 5, narrowing the gap to DeepSeek-Math-V2 under the same grader. Details of evaluation framework can be found in Appendix F.

4.2 IOI 2025, ICPC World Finals 2025, and Competitive Programming

We evaluate Nemotron-Cascade-2-30B-A3B on olympiad-level competitive programming tasks using test-time scaling. For IOI 2025, we adapt the IOI pipeline from Nemotron-Cascade (Wang et al., 2026) into a multi-round *generate-select-submit* framework that incorporates official judge

Table 4: IMO-ProofBench performance. Expert-evaluated results are taken from the leaderboard (accessed on 2026/3/9).

Model	IMO-ProofBench		
	Basic	*Advanced*	*Overall*
Aletheia (Feng et al., 2026)	-	91.9	-
Gemini 3 Deep Think (Gemini Team, 2026)	-	76.7	-
Gemini Deep Think (IMO Gold) (Gemini Team, 2025b)	89.0	65.7	76.7
DeepSeek-Math-V2-671B-A37B (Shao et al., 2025)	99.0	61.9	80.2
DeepSeek-Math-V2-671B-A37B †	99.5	57.7	78.6
Nemotron-Cascade-2-30B-A3B†	92.5	53.4	72.9
GPT-5.2-Thinking (high) (OpenAI, 2025)	-	35.7	-
Gemini 3 Pro (Gemini Team, 2025a)	-	30.0	-
GPT-5 Pro (OpenAI, 2025)	-	28.6	-

† Uses DeepSeek-V3.2-Speciale as judge with the LLM ProofAutoGrader (Luong et al., 2025).

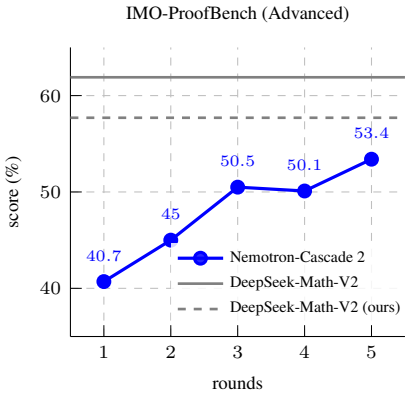


Figure 3: Test-time scaling enables smaller model to closer the gap.

Table 5: Competitive programming results on comprehensive benchmarks, evaluated against a significantly expanded set of proprietary and open-source baseline models.

Models	LiveCodeBench v6 2408 - 2505	LiveCodeBench Pro 25Q1 Easy	25Q1 Med	25Q1 Hard	25Q2 Easy	25Q2 Med	25Q2 Hard	Codeforces 2501 - 2507 ELO	Codeforces Percentile
GPT-5.2 (high)	-	96.6	75.0	5.9	91.8	59.6	23.1	2590	99.9
Gemini-3 Pro	90.7	94.4	70.0	5.9	94.8	45.6	7.7	2440	99.8
GPT-o4-mini (high)	80.2	85.4	51.7	0.0	84.5	29.8	0.0	2266	99.5
DeepSeek-v3.2-Speciale	88.7	89.7	48.1	0.0	88.5	43.1	0.0	2353	99.7
GPT-OSS-120B (high)	87.0	88.8	41.9	0.7	88.5	31.1	0.0	2320	99.6
Kimi-K2.5-1T-thinking	85.0	88.5	45.6	0.0	90.2	37.9	0.0	2333	99.7
Qwen-3.5-397B-A17B	83.6	89.3	44.4	0.0	88.1	31.4	0.0	2350	99.7
Nemotron-3-Super-120B-A12B	78.7	83.0	31.0	0.0	81.7	23.2	0.0	2212	99.4
Qwen3-235B-A22B-Thinking-2507	78.7	75.8	18.8	0.0	77.6	17.5	0.0	2119	98.6
Nemotron-Cascade-14B	74.6	71.6	16.3	0.0	68.9	10.5	0.0	2004	97.9
Nemotron-Cascade-2-30B-A3B	**87.2**	**88.1**	**39.2**	**0.7**	**87.0**	**27.6**	0.0	**2320**	**99.6**
Nemotron-Cascade-2-30B-A3B (TIR)	**88.4**	**91.0**	**45.2**	**2.2**	**89.3**	**36.8**	0.0	**2345**	**99.7**

feedback and transfers insights across subtasks. Despite its compact 30B-A3B scale, the model achieves a gold-medal score of 439.28 within at most $40 \times 50 = 2000$ generations, with 507.66 attainable within 5000 generations; it fully solves Problems 3 and 4 and reaches over 86 points on the heuristic-optimization Problem 2 within only 5 rounds. On ICPC World Finals 2025, the model solves 10 out of 12 problems, corresponding to a #4 Gold-medal placement, with 8 problems solved within 100 submissions. Beyond these contest settings, Table 5 shows that Nemotron-Cascade-2-30B-A3B achieves 87.2 on LiveCodeBench v6 and a simulated Codeforces ELO of 2320, competitive with substantially larger open-source models; enabling Tool-Integrated Reasoning further improves performance to 88.4 on LiveCodeBench v6 and 2345 ELO. Together, these results show that inference-time scaling and tool use can unlock strong competitive-programming performance from a comparatively compact model.

5 Conclusion

In this work, we introduced Nemotron-Cascade-2-30B-A3B that combines strong reasoning and agentic capabilities. By grouping compatible domains within Cascade RL stages and incorporating multi-domain on-policy distillation, our training pipeline improves domain coverage and training efficiency while recovering performance lost to cross-domain interference. The resulting model achieves gold-medal-level performance on IMO 2025 and IOI 2025, demonstrating that compact MoE models can approach frontier reasoning performance through effective post-training. We release the model weights and training data to support further open research.

AI USE STATEMENT

In this work, we used generative AI tools solely to correct grammatical errors in the manuscript and for no other purpose.

REPRODUCIBILITY STATEMENT

We open-sourced the model checkpoints for evaluation, the SFT and RL datasets for training, and detailed hyperparameters for reproducing all training runs described in this work.

REFERENCES

Gretel synthetic safety alignment dataset, 12 2024. URL https://huggingface.co/datasets/gretelai/gretel-safety-alignment-en-v1.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

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*, 2025.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. *arXiv preprint arXiv:2505.20411*, 2025.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-Nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025a.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025b.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *Advances in neural information processing systems*, 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.

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.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, et al. Cl-bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*, 2026.

Wei Du, Shubham Toshniwal, Branislav Kisacanin, Sadegh Mahdavi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. Nemotron-math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision. *arXiv preprint arXiv:2512.15489*, 2025.

Tony Feng, Trieu H. Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang-hyun Kim, Federico Pasqualotto, Sergei Gukov, Jonathan N. Lee, Junsu Kim, Kaiying Hou, Golnaz Ghiasi, Yi Tay, YaGuang Li, Chenkai Kuang, Yuan Liu, Hanzhao Lin, Evan Zheran Liu, Nigamaa Nayakanti, Xiaomeng Yang, Heng-Tze Cheng, Demis Hassabis, Koray Kavukcuoglu, Quoc V. Le, and Thang Luong. Towards autonomous mathematics research. *arXiv preprint arXiv:2602.10177*, 2026. doi: 10.48550/arXiv.2602.10177.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are we done with mmlu?, 2024.

Gemini Team. A new era of intelligence with gemini 3. `https://blog.google/products-and-platforms/products/gemini/gemini-3/`, 2025a. Google Blog, November 18, 2025.

Gemini Team. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/, 2025b. Google DeepMind Blog, July 21, 2025.

Gemini Team. Gemini 3 deep think: Advancing science, research and engineering. `https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-deep-think/`, 2026. Google Blog, February 12, 2026.

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 *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, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: On-policy distillation of large language models. In *ICLR*, 2024.

Nuno M Guerreiro, Ricardo Rei, Daan van Stigt, Luisa Coheur, Pierre Colombo, and André FT Martins. xcomet: Transparent machine translation evaluation through fine-grained error detection. *Transactions of the Association for Computational Linguistics*, 12:979–995, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Adib Hasan, Ileana Rugina, and Alex Wang. Pruning for protection: Increasing jailbreak resistance in aligned llms without fine-tuning. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 417–430, 2024.

Zhongmou He, Yee Man Choi, Kexun Zhang, Jiabao Ji, Junting Zhou, Dejia Xu, Ivan Bercovich, Aidan Zhang, and Lei Li. Hardtests: Synthesizing high-quality test cases for llm coding. *arXiv preprint arXiv:2505.24098*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

HMMT. Harvard-mit mathematics tournament february 2025, 2025.

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.

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

IMO. International Mathematical Olympiad, 2025.

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.

Gregory Kamradt. Needle in a haystack - pressure testing llms. *Github*, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.

Kimi-Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi K1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. AceMath: Advancing frontier math reasoning with post-training and reward modeling. *ACL*, 2024b.

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. *ICLR*, 2026.

LM-Provers, Yuxiao Qu, Amrith Setlur, Jasper Dekoninck, Edward Beeching, Jia Li, Ian Wu, Lewis Tunstall, and Aviral Kumar. Qed-nano: Teaching a tiny model to prove hard theorems. https://huggingface.co/spaces/lm-provers/qed-nano-blogpost, 2026. Blog post.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

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. *arXiv preprint arXiv:2404.03027*, 2024.

Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, et al. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35406–35430, 2025.

Wenjie Ma, Andrei Cojocaru, Neel Kolhe, Bradley Louie, Robin Said Sharif, Haihan Zhang, Vincent Zhuang, Matei Zaharia, and Sewon Min. Reliable fine-grained evaluation of natural language math proofs. *arXiv preprint arXiv:2510.13888*, 2025.

MAA. American Invitational Mathematics Examination - AIME 2025, 2025.

MAA. American Invitational Mathematics Examination - AIME 2026, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

NVIDIA. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025a.

NVIDIA. Nemotron-3-Nano-30B-A3B-Base-BF16, 2025b. URL https://huggingface.co/nvidia/NVIDIA-Nemotron-3-Nano-30B-A3B-Base-BF16.

NVIDIA. Nemo gym: An open source library for scaling reinforcement learning environments for llm. https://github.com/NVIDIA-NeMo/Gym, 2025c. GitHub repository.

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

NVIDIA. NeMo-Skills. https://github.com/NVIDIA-NeMo/Skills, 2025. GitHub repository.

OpenAI. Introducing SWE-bench Verified, 2024.

OpenAI. Introducing GPT-5, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *NeurIPS*, 35, 2022.

Jiayi Pan*, Xingyao Wang*, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In *ICML*, 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 *ICML*, 2025.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, and et al. Humanity's last exam, 2025. URL https://arxiv.org/abs/2501.14249.

Renjie Pi, Grace Lam, Mohammad Shoeybi, Pooya Jannaty, Bryan Catanzaro, and Wei Ping. On data engineering for scaling llm terminal capabilities, 2026. URL https://arxiv.org/abs/2602.21193.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following, 2025. URL https://huggingface.co/datasets/allenai/IF_multi_constraints_upto5.

Qwen-Team. Qwen3-235B-A22B-Thinking-2507 , 2025. URL https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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

Zhihong Shao, Yuxiang Luo, Chengda Lu, ZZ Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *arXiv preprint arXiv:2511.22570*, 2025.

Atefeh Sohrabizadeh, Jialin Song, Mingjie Liu, Rajarshi Roy, Chankyu Lee, Jonathan Raiman, and Bryan Catanzaro. Nemotron-cortexa: Enhancing llm agents for software engineering tasks via improved localization and solution diversity. In *Forty-second International Conference on Machine Learning*, 2025.

Artificial Analysis Team. Artificial analysis long context reasoning benchmark(lcr), 2025.

Minyang Tian, Luyu Gao, Shizhuo D Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650, 2024.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-Cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *NeurIPS*, 2026.

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. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=OJd3ayDDoF.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

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, 2025b. URL https://arxiv.org/abs/2505.11475.

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, 2025c. URL https://arxiv.org/abs/2505.11475.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Peng Xu, Wei Ping, Xianchao Wu, Chejian Xu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa 2: Bridging the gap to proprietary llms in long context and rag capabilities. *arXiv preprint arXiv:2407.14482*, 2024.

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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1513–1532, 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. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Zonghan Yang, Shengjie Wang, Kelin Fu, Wenyang He, Weimin Xiong, Yibo Liu, Yibo Miao, Bofei Gao, Yejie Wang, YINGWEI MA, Yanhao Li, Yue Liu, Zhenxing Hu, kaitai zhang, Shuyi Wang, Huarong Chen, Flood Sung, Yang Liu, Yang Gao, Zhilin Yang, and Tianyu Liu. Kimi-dev: Agentless training as skill prior for SWE-agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tYppHuGhxJ.

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.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chengxing Xie, Cunxiang Wang, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

Zihan Zheng, Zerui Cheng, Zeyu Shen, Shang Zhou, Kaiyuan Liu, Hansen He, Dongruixuan Li, Stanley Wei, Hangyi Hao, Jianzhu Yao, et al. Livecodebench pro: How do olympiad medalists judge llms in competitive programming? *arXiv preprint arXiv:2506.11928*, 2025.

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

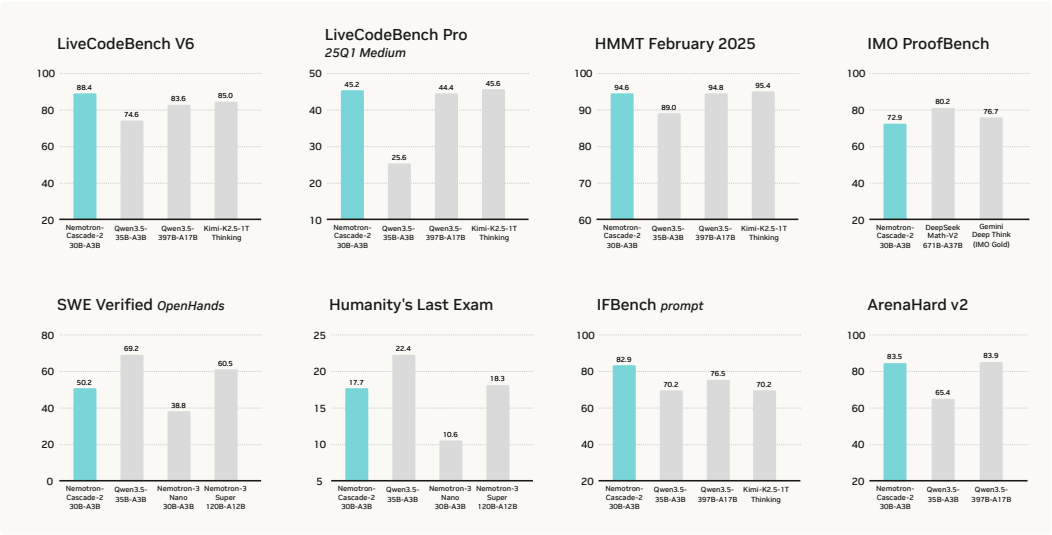


Figure 4: Overview of Nemotron-Cascade 2 performance.

A Related Work

RL post-training for LLMs. Reinforcement learning has become a central tool for LLM post-training, including RLHF for preference alignment (Ouyang et al., 2022; Bai et al., 2022) and RL with verifiable rewards for reasoning, coding, and tool-use tasks (Shao et al., 2024; Guo et al., 2025; Kimi-Team et al., 2025; Yang et al., 2025; Yu et al., 2025). Recent reasoning systems often train on heterogeneous domains within broad RL stages, improving coverage but making optimization sensitive to task mixture, reward scale, verifier cost, and response-length dynamics. Cascade RL (Wang et al., 2026) instead decomposes post-training into domain-aware stages to improve stability and reduce interference. Cascade-2 extends this framework with in-stage multi-domain RL, jointly training compatible domains while retaining staged control over cross-domain interference.

On-policy Distillation. Distillation is widely used to obtain compact reasoning models, either through teacher-generated traces or stronger checkpoints used during later optimization (Bercovich et al., 2025; NVIDIA, 2025a; Ahmad et al., 2025; Yang et al., 2025; Lu & Lab, 2025; Gu et al., 2024). Multi-domain On-policy Distillation (MOPD) is closest to the latter line, but differs from standard off-policy SFT-style distillation because the student samples its own trajectories and receives dense token-level advantages from the strongest intermediate domain teachers (Xiao et al., 2026; Zeng et al., 2026). In Cascade-2, MOPD serves as a stabilization stage after multi-domain RL, recovering math regressions caused by response-length compression and improving the initialization for downstream RLHF and agentic RL.

B Details of Supervised Fine-Tuning

In this section, we describe the training framework and data curation process for supervised fine-tuning (SFT), the first stage of our post-training pipeline. This stage equips the model with foundational capabilities, including reasoning, conversational ability, instruction following, and agentic and software engineering skills.

B.1 Training Framework

B.1.1 Overview

Our SFT data spans a broad range of domains, including mathematics, coding, science, tool use, agentic tasks, and software engineering, as well as more general domains such as multi-turn dialogue,

Chat Template

<|im_start|>system
You are a helpful and harmless assistant.
You are not allowed to use any tools.<|im_end|>
<|im_start|>user
Write a paragraph about San Francisco. <|im_end|>
<|im_start|>assistant
<think></think>San Francisco, located on the northern coast of California, is a vibrant and iconic city known for its stunning Golden Gate Bridge, steep hills, and eclectic neighborhoods like ..... <|im_end|>
<|im_start|>user
What should I prepare for a summer trip to San Francisco? <|im_end|>
<|im_start|>assistant
<think>
Okay, the user is asking about preparing for a summer trip to San Francisco. They've already seen my description of the city, so they're likely planning their visit.
Hmm, summer in SF is famously unpredictable with microclimates and fog, so the key advice should emphasize layers and rain gear. The user might not realize how drastically temperatures can change between neighborhoods or elevations.
I should also highlight practical tips they might overlook: comfy shoes for hills, hydration due to dry air, and booking reservations since summer is peak season. .....
</think>
Here's what to prepare for a summer trip to San Francisco, focusing on essentials and maximizing your experience: ..... <|im_end|>
<|im_start|>user
Tell me more about it. <|im_end|>
<|im_start|>assistant
<think></think>A summer trip to San Francisco can be amazing, but the experience is a bit different from typical summer destinations. Here are some additional things that will help you plan better: ..... <|im_end|>

System Prompt for Tool Calling

<|im_start|>system
You are a customer service agent that helps the user. The policy that determines how you should respond to requests from users is described below between <policy> and </policy> tags.
<policy> ..... </policy>
Try to be helpful and always follow the policy.
 # Tools
You have access to the following functions:
<tools>
<function>
<name>function-name</name>
<description>function-description</description>
<parameters>
<parameter>
<name>parameter-name</name>
<type>parameter-type</type>
<description>parameter-description</description>
<title>parameter-title</title>
</parameter>
..... *(more parameters to add for this function)*
</parameters>
</function>
..... *(more functions to add in the tool list)*
</tools>
If you choose to call a function, ONLY reply in the following format with NO suffix:
<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
</function>
</tool_call><|im_end|>
<|im_start|>user
Book a hotel in San Francisco. <|im_end|>

Figure 5: (Left) The chat template uses adjacent *<think></think>* tokens to indicate non-thinking mode, and a single *<think>* followed by *\n* to indicate thinking mode. (Right) For tool calling, the available tools are listed in the system prompt. The model is instructed to call tools within the *<tool_call>* and *</tool_call>* tags.

knowledge-intensive question answering, creative writing, role-playing, safety, and instruction following.

We pack all SFT samples into sequences of up to 256K tokens and train the model in a single stage. Empirically, we find that the SFT model reaches optimal performance after approximately 1.5 epochs. The SFT training hyperparameters can be found in Appendix H.

B.1.2 Chat Template

Our chat template is depicted in Figure 5. There are two changes to the chat template compared with Nemotron-Cascade (Wang et al., 2026). First, we remove the `/think` and `/no_think` tags for simplicity. Second, we prepend an empty `<think></think>` block to activate the non-thinking mode.

For tool calling task, we specify all available tools in the system prompt within the `<tools>` and `</tools>` tags, and instruct the model to perform tool calls wrapped within the `<tool_call>` and `</tool_call>` tags.

B.2 SFT Data Curation

B.2.1 Math

Our non-proof math prompts are primarily sourced from Nemotron-Cascade (Wang et al., 2026) and Nemotron-Math-v2 (Du et al., 2025), from which we collect 1.8M tool calling (i.e., python) samples and 1.9M non-tool samples, with responses generated by DeepSeek-V3.2 and DeepSeek-V3.2-Speciale (Liu et al., 2025), respectively. In addition, we collect 676K samples from the generation-selection category (without tool calling) of Nemotron-3-Nano (Blakeman et al., 2025a), with responses generated by GPT-OSS-120B (Agarwal et al., 2025). In total, the competition math SFT comprises 1.8M tool-calling samples and 2.6M samples without tool use.

For mathematical natural language proof, we collect 98K mathematical proof problems from the AOPS split of Nemotron-Math-Proofs-v1 (Du et al., 2025). We generate multiple samples per problem to cover two capabilities including proof generation (410K) and proof verification (400K) using DeepSeek-V3.2-Speciale (Liu et al., 2025), resulting in a total of 816K samples.

B.2.2 CODE REASONING

Built on Nemotron-Cascade 1 (Wang et al., 2026), we curate approximately 165K unique coding prompts from several open-source datasets, including OpenCode-Stage2 (Huang et al., 2024), OpenCodeReasoning (Ahmad et al., 2025), and HardTests (He et al., 2025). These prompts are originally sourced from competitive programming platforms such as Codeforces, AtCoder, AIZU, and CodeChef. To encourage prompt diversity and reduce redundancy in our SFT training set, we apply strict deduplication using two methods: (1) sample I/O fingerprinting and (2) n-gram-based text analysis. This process removes approximately 24.2% of self-duplicated coding prompts.

We choose GPT-OSS-120B (Agarwal et al., 2025) as our SFT teacher model due to its strong code reasoning capabilities. For each coding prompt with verifiable test cases, we apply correctness filtering to the teacher's reasoning traces, retaining only those that generate correct code. For prompts without verifiable test cases, we generally select longer reasoning traces under the assumption that they reflect more thorough problem analysis. This pipeline yields a final dataset comprising 1.9M Python reasoning traces, 1.0M C++14 reasoning traces, and 1.3M Python tool-calling reasoning traces for competitive coding.

Scientific Coding: We further collect scientific research coding prompts spanning the domains of biology, material science, physics, chemistry, and mathematics. The responses to these prompts are generated by GPT-OSS-120B (Agarwal et al., 2025), resulting in a total of 1.1M SFT samples.

B.2.3 SCIENCE

The science prompts we collect span physics, chemistry, and biology. We use 1.4M science SFT samples from Nemotron-Cascade (Wang et al., 2026) and an additional 1.3M samples from Nemotron-3-Nano (Blakeman et al., 2025a). Responses in both datasets are generated by GPT-OSS-120B (Agarwal et al., 2025).

B.2.4 LONG CONTEXT

We adopt the 160K long context SFT data from Nemotron-3-Nano (Blakeman et al., 2025a), which has an average sequence length of 128K tokens. In addition, we collect another 74K long context SFT from ChatQA-2 (Xu et al., 2024), which has an average length of 29K tokens.

B.2.5 GENERAL CHAT

We source prompts from Nemotron-Cascade 1 (Wang et al., 2026) and construct 4.9M reasoning-on and 372K reasoning-off samples. Responses for reasoning-on samples are generated by GPT-OSS-120B (Agarwal et al., 2025). For reasoning-off samples, 300K responses are drawn from high-quality annotated short answers within the dataset itself, while an additional 330K are generated by DeepSeek-V3-0324 (Liu et al., 2024a) to improve response quality.

To enhance multi-turn dialogue capabilities, we synthesize approximately 700K multi-turn conversation samples using two GPT-OSS-120B (Agarwal et al., 2025) instances in a role-playing setup, where one instance plays the user and the other the assistant. The user-side model may terminate the conversation at any point to prevent repetitive exchanges.

We additionally incorporate 4.6M reasoning-on chat samples from Nemotron-3-Nano (Blakeman et al., 2025a), with prompts drawn from LMSYS (Zheng et al., 2023) and WildChat (Zhao et al., 2024). Responses are generated by GPT-OSS-120B (Agarwal et al., 2025), Qwen3-235B-A22B-Thinking-2507, and Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025).

B.2.6 INSTRUCTION FOLLOWING

We source prompts from Nemotron-Cascade 1 (Wang et al., 2026) and generate approximately 230K reasoning-on responses using GPT-OSS-120B (Agarwal et al., 2025) and 64K reasoning-off responses using DeepSeek-V3-0324 (Liu et al., 2024a). In addition, we incorporate 497K instruction-following samples from Nemotron-3-Nano (Blakeman et al., 2025a), including 457K reasoning-on and 40K reasoning-off responses. These responses are generated by GPT-OSS-120B (Agarwal et al., 2025), Qwen3-235B-A22B-Thinking-2507, and Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025).

B.2.7 SAFETY

We collect 4K safety SFT samples from Nemotron-3-Nano (Blakeman et al., 2025a) to enable models to exhibit appropriate refusal behavior when encountering unsafe inputs. The SFT prompts are originally sourced from Nemotron Content Safety v2 (Ghosh et al., 2025), Gretel Safety Alignment v1 (gre, 2024), Harmful Tasks (Hasan et al., 2024), and Red-Team-2K (Luo et al., 2024).

B.2.8 CONVERSATIONAL AGENT

Aside from the Python tool-use data for math and code reasoning, we further gather tool-use samples in multi-turn conversational settings, where multiple tools are available and the assistant must determine which tools to invoke and how to use them effectively. We collect 822K conversational tool-use samples from Nemotron-3-Nano (Blakeman et al., 2025a), with responses generated by Qwen3-235B-A22B-Thinking-2507, Qwen3-32B, Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025), and GPT-OSS-120B (Agarwal et al., 2025).

B.2.9 SOFTWARE ENGINEERING AGENT

We curate the software engineering (SWE) data using agentic scaffolds, including OpenHands (Wang et al., 2025a), SWE-Agent (Yang et al., 2024), Mini-SWE-Agent, and the agentless scaffold proposed by Wei et al. (2025), to enhance the models' agentic software engineering capabilities. First, we utilize the data from Nemotron 3 Nano (Blakeman et al., 2025a) and Super (Blakeman et al., 2025b), which includes SWE agentic trajectories generated using Qwen3-Coder-480B-A35B-Instruct (Yang et al., 2025). The problem instances are drawn from SWE-Gym (Pan* et al., 2025), SWE-rebench (Badertdinov et al., 2025), and R2E-Subset (Jain et al., 2025). Second, we employ SWE agentless data from Nemotron-Cascade 1 (Wang et al., 2026), which includes three main tasks: (1) buggy code localization, (2) code repair, and (3) test case generation. Following the established procedure in Wang et al. (2026), we reconstruct the code repair data using DeepSeek-V3.2 (Liu et al., 2025).

Our preliminary study shows that incorporating SWE agentless data improves models' effectiveness on SWE agentic tasks. For example, fine-tuning solely on agentic data achieves Pass@1 of 48.9 and Pass@4 of 62.8, whereas fine-tuning on a combination of agentic and agentless data improves performance to Pass@1 of 49.9 and Pass@4 of 65.2 on SWE-bench Verified using OpenHands. Based on this observation, we combine 125K agentic samples and 389K agentless samples as the supervised fine-tuning (SFT) data for SWE tasks. Our models are trained in non-thinking mode on SWE agentic data and in thinking mode on SWE agentless data.

B.2.10 TERMINAL AGENT

To enhance agentic capabilities for terminal use, we adopt the *Terminal-Task-Gen* methodology (Pi et al., 2026) to curate our training tasks. This framework consists of (1) dataset adapters that transform static data into interactive terminal formats, and (2) synthetic tasks generated from both diverse seed prompts and a structured terminal skill taxonomy. Using this framework, we curate 490K samples in total. Specifically, we first adapt 162K math, 32K code, and 32K SWE-specific samples from existing high-quality sources (Wang et al., 2026), which establishes broad foundational coverage. To further improve targeted skill refinement, we synthesize 120K seed-based and 140K skill-based tasks. For trajectory construction, we leverage the tasks curated from above, and employ DeepSeek-V3.2 (Liu et al., 2025) as the core engine to generate step-by-step solution traces via an execution-feedback loop within isolated Docker environments. The Terminus 2 agent framework (Merrill et al., 2026) serves as the underlying scaffolding and tool-use protocol, enabling the model to interact with the terminal and complete complex tasks.

C Details of Cascade RL and Multi-Domain On-Policy Distillation

Following a similar approach to Nemotron-Cascade 1 (Wang et al., 2026), we apply Cascaded Reinforcement Learning (Cascade RL) as our post training pipeline. In particular, we integrated the Multi-Domain On-Policy Distillation (MOPD) along the Cascade RL process.

C.1 Training Framework

We illustrate our training process in Figure 1. In this work, we start the Cascade RL process with IF-RL (§C.2) to establish foundational instruction adherence, followed by multi-domain RL (§C.3) to enhance the model's tool-calling capabilities, STEM reasoning, and response format adherence. We then transition to Multi-domain On-policy Distillation (§C.4) to unify specialized expertise into a single, cohesive policy to mitigate performance degradation. We continue with RLHF (§C.5) for human alignment, Long-context RL (§C.6) to enhance reasoning over massive input sequences, Code RL (§C.7) for competitive coding problems, and finally SWE RL (§C.8) for mastering agentic software interactions.

C.1.1 RL Training Configuration

Throughout the entire Cascade RL process, we use Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024) with strict **on-policy** training following Nemotron Cascade (Wang et al., 2026). We adopt on-policy training for improved stability and higher accuracy. We conduct our training using the Nemo-RL repository (NVIDIA, 2025d).

At each iteration, we generate a group of G rollouts from the current policy π_θ and then perform a *single* gradient update. This ensures that the policy used for data collection always matches the one being updated, making the importance sampling ratio exactly 1. This on-policy setup contributes to stable RL training and mitigates entropy collapse. In addition, we remove KL divergence term entirely, which simplifies the GRPO objective to the standard ***REINFORCE*** objective (Williams, 1992) with group-normalized rewards and token-level loss (Yu et al., 2025):

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\ \{o_i\}_{i=1}^{G}\sim\pi_\theta(\cdot|q)}\left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\hat{A}_{i,t}\right],$$
$$\text{where}\quad \hat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^{G})}{\text{std}(\{r_i\}_{i=1}^{G})}\quad \text{for all } t, \tag{4}$$

and $\{r_i\}_{i=1}^{G}$ denotes the group of G rewards assigned to the sampled responses $\{o\}_{i=1}^{G}$ for a given question q drawn from the dataset $\mathcal{D}$, verified against the ground-truth answer a in RLVR. For RLHF, r_i is the aggregated reward score from the generative reward model for response o_i and question q. Details of the reward functions for different domains will be provided in the corresponding subsections.

C.2 Instruction-Following Reinforcement Learning (IF-RL)

In this subsection, we describe our instruction-following RL recipe, which serves as the first stage of our Cascade RL. We demonstrate that applying verifiable IF-RL significantly improves instruction adherence, achieving a state-of-the-art accuracy of 83.13% on IFBench (Pyatkin et al., 2025).

C.2.1 Dataset

We use the same instruction-following training data used for NVIDIA Nano-v3 post-training (Blakeman et al., 2025a). The instructions in this dataset are designed for objective verifiability, for instance, requiring a response to be under 200 words. This making the dataset well-suited for training and evaluating models on strict adherence. Given the high baseline quality of the data, our curation process mainly resolves formatting inconsistencies within the keyword arguments for certain instruction types (e.g., `count_increment_word`).

C.2.2 TRAINING RECIPE

Following (Wang et al., 2026), we also apply dynamic filtering (Yu et al., 2025). This technique filters out samples where all rollouts are either entirely correct or entirely incorrect. By ensuring that every prompt in a batch provides effective gradients, dynamic filtering stabilizes IF-RL training and pushes the upper bound of model performance. Furthermore, we observed that extended IF-RL training can lead to excessive token usage, which is often unnecessary for fulfilling specific constraints in general chat domains. To mitigate this, we apply overlong penalty, which penalizes samples that fail to complete generation within the maximum sequence length with a zero reward.

Unlike Nemotron Cascade (Wang et al., 2026), we position IF-RL as the first stage of our Cascade RL training for two primary reasons: (i) IF-RL can negatively impact human alignment capabilities (e.g., ArenaHard), while our subsequent generative-reward-model-based RLHF has a negligible impact on instruction following scores. By prioritizing instruction adherence first, we can focus on maximizing instruction following performance and then utilize the later stages to recover and refine human preference alignment. (ii) An early IF-RL stage produces a model with superior instruction-following capabilities, which serves as a strong teacher for subsequent multi-domain on-policy distillation. Another difference from Nemotron Cascade (Wang et al., 2026) is that our IF-RL is trained exclusively in "thinking mode" without incorporating a reward model. We found that the "thinking mode" yields higher accuracy on instruction-following benchmarks (e.g., IFBench (Pyatkin et al., 2025)). Because subsequent RL stages recover any regressions in human preference alignment introduced during IF-RL, we can focus entirely on maximizing instruction adherence without incurring the computational overhead of an auxiliary reward model.

We use a batch size of 128, sampling 16 responses per prompt with temperature 1.0 and top-p 1.0. We adopt a learning rate of 3e-6 with AdamW (Kingma & Ba, 2014), and set both the entropy loss coefficient and KL loss coefficient to 0. Our IF-RL with dynamic filtering takes around 180 steps. The full set of hyperparameters is provided in Appendix H.

C.3 MULTI-DOMAIN RL

Following IF-RL, we conduct an additional stage of multi-domain RL that covers three capabilities: multi-choice question answering (MCQA) in the STEM domain, agentic tool calling, and structured output for instruction following. The datasets are drawn from the NVIDIA Nano-v3 RL training blend (Blakeman et al., 2025a). The data mixture consists of approximately 55% MCQA, 30% agentic tool calling using the Workplace Assistant setup (Blakeman et al., 2025a), and 15% structured output.

We group these domains into a single multi-domain RL stage for two main reasons. First, we do not observe performance degradation across evaluation benchmarks when training on the blended domains. Instead, the model exhibits consistent improvements on benchmarks including MMLU-Pro, τ^2-Bench, and IF-Bench. Second, the response lengths and verification times of these datasets are similar, which minimizes training inefficiencies caused by waiting for longer generations or slower environment verification.

During training, we use a batch size of 128 and sample 16 responses per prompt with temperature 1.0 and top-p 1.0 (see Appendix H). We adopt a learning rate of 3×10^{-6} with AdamW (Kingma & Ba, 2014), and set both the entropy loss coefficient and KL loss coefficient to zero. This multi-domain RL stage runs for approximately 70 training steps.

C.4 MULTI-DOMAIN ON-POLICY DISTILLATION (MOPD)

This section gives the full MOPD objective, teacher and data setup, and hyperparameters for §3.5.

Objective. Let π^{inf} denote the student policy used for generation in the inference engine and π^{train} the student policy optimized by the training engine. For each prompt x, we sample $y = (y_1, \ldots, y_T) \sim \pi^{\text{inf}}(\cdot \mid x)$ and select the teacher π^{domain} associated with the prompt's domain. Writing $s_t = (x, y_{<t})$ for the decoding state, the token-level advantage in Eq. (3) is $a_t^{\text{MOPD}} = \log \pi^{\text{domain}}(y_t \mid s_t) - \log \pi^{\text{train}}(y_t \mid s_t)$, the negative of the single-sample reverse KL estimate at step t. It is positive when the teacher assigns a higher probability to the sampled token than the student, and it approaches zero as the student matches the teacher. It is computed only on the sampled token rather than over the full vocabulary.

Because responses are sampled under π^{inf} but optimized under π^{train}, we apply truncated importance weighting to account for the train–inference mismatch:

$$r_t = \frac{\pi^{\text{train}}(y_t \mid s_t)}{\pi^{\text{inf}}(y_t \mid s_t)}, \qquad w_t = \text{sg}[r_t]\,\mathbf{1}[\epsilon_{\text{low}} \le r_t \le \epsilon_{\text{high}}]\,, \tag{5}$$

where $\text{sg}[\cdot]$ denotes stop-gradient. We then optimize the surrogate objective

$$\mathcal{L}_{\text{MOPD}} = -\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi^{\text{inf}}(\cdot|x)}\left[\frac{1}{|\mathcal{V}(y)|}\sum_{t\in\mathcal{V}(y)} w_t\,\text{sg}\big[a_t^{\text{MOPD}}\big]\log \pi^{\text{train}}(y_t \mid s_t)\right], \tag{6}$$

where $\mathcal{V}(y)$ is the set of valid response tokens retained by the token mask.

Teachers and data. The math teacher is the SFT checkpoint, which already exhibits strong mathematical reasoning. The RLHF teacher is obtained by applying RLHF to the SFT checkpoint. The multi-domain teacher is selected from checkpoints of the IF-RL + multi-domain RL stages. Prompts are sampled from the corresponding RL training pools (RLHF, IF-RL, and multi-domain RL), together with AceReason-Math (Chen et al., 2025) for math. Teacher assignment is a predefined rule based on the prompt's source pool: math prompts are scored by the math teacher, RLHF prompts by the RLHF teacher, and IF-RL and multi-domain prompts by the multi-domain teacher. Because every prompt originates from exactly one pool, each response is scored by a single teacher and no per-token routing is involved; the teachers interact only through the shared parameter update within a batch. When one teacher is strong across several domains, we merge those domains and distill from that teacher alone.

Why teachers from the cascade. An alternative is to train a separate expert per domain and run a single MOPD stage at the end. We instead select teachers from checkpoints along the cascade for two reasons: training dedicated experts becomes costly as the number of domains grows, and intermediate checkpoints stay close to the student in parameter space, which makes distillation more effective and stable.

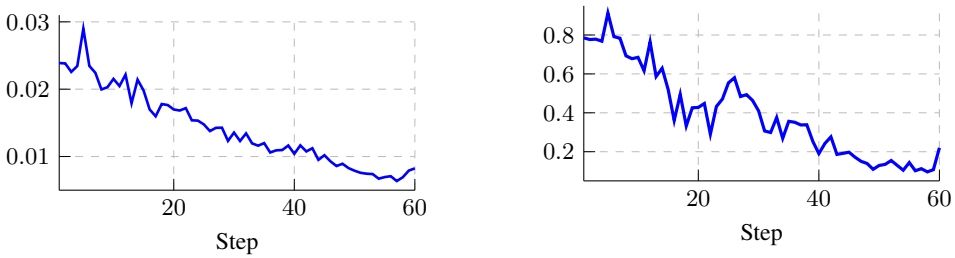
On-policy sampling. MOPD is strictly on-policy, with one update per batch. Responses are generated by a separate inference engine that holds a snapshot of the current student weights; π^{inf} refers to this snapshot, and the resulting train–inference mismatch is corrected with the truncated importance weights in Eq. (5).

Convergence on ArenaHard v2. Table 6 compares MOPD and RLHF when both start from the IF-RL + multi-domain RL checkpoint.

Table 6: ArenaHard v2 starting from the IF-RL + multi-domain RL checkpoint. MOPD reaches a higher Hard Prompt score in 52 steps than RLHF does in 160 steps, and RLHF after MOPD gives the best result on both subsets.

Method	Steps	Hard	Creative
Initial	0	70.4	39.7
RLHF	160	80.7	71.2
MOPD	52	85.5	71.0
MOPD → RLHF	52 + 25	**87.5**	**80.3**

Hyperparameters. Unless otherwise specified, we use 128 prompts per update with 4 rollouts per prompt (512 responses per batch). In later experiments, 512 prompts with 1 rollout each gives slightly more stable optimization with similar final results. The learning rate is 2×10^{-6}, linearly warmed up from 2×10^{-7} over the first 30 steps. Warm-up is important for stability: gradient norms are large early in training and drop quickly afterward (Figure 6(b)). Training typically converges within 40–50 steps (Figure 6(a)). For truncated importance weighting, we set $\epsilon_{\text{low}} = 0.5$ and $\epsilon_{\text{high}} = 2.0$.



(a) Reverse KL (b) Gradient norm

Figure 6: MOPD training dynamics. (a) Token-level reverse KL between student and teachers decreases steadily. (b) Gradient norm is large during warm-up and decreases afterward.

C.5 Reinforcement Learning from Human Feedback (RLHF)

Building on multi-domain on-policy distillation, our RLHF recipe focuses on human preferece learning. This process further enhances creative writing and non-verifiable problem-solving in coding and mathematics, as measured by ArenaHard v2 (Li et al., 2024), while maintaining performance across other domains without degradation.

C.5.1 Dataset

We adopt the RLHF training dataset from NVIDIA Nano-v3 (Blakeman et al., 2025a), which comprises HelpSteer3 (Wang et al., 2025b), a commercially-friendly subset of the arena-human-preference-140k dataset (Chiang et al., 2024), and a synthetic safety blend (Blakeman et al., 2025a). Following the NVIDIA Nano-v3 (Blakeman et al., 2025a), we utilize Qwen3-235B-A22B-Thinking-2507 (Yang et al., 2025) as our generative reward model (GenRM), trained via the HelpSteer3 framework (Wang et al., 2025c). Given a conversation history, a user request, and two candidate responses, the GenRM first reasons through the strengths and weaknesses of each response before producing individual helpfulness scores and a final comparative ranking.

C.5.2 Training recipe

Following a training recipe similar to NVIDIA Nano-v3 (Blakeman et al., 2025a), we conduct RLHF using the GenRM. To ensure the training signals are of high quality, we adopt pair-wise comparisons for all pairs of rollouts per prompt. We aggregate the reward scores in the same way as NVIDIA Nano-v3 RLHF training, and apply the same length-normalized reward adjustment and quality-gated conciseness bonus (Blakeman et al., 2025a). These mechanisms encourage shorter responses without sacrificing quality, effectively mitigating the rapid growth of inference token usage.

Different from Nemotron Cascade (Wang et al., 2026), we train RLHF exclusively in the thinking mode. While incorporating both thinking and non-thinking modes can improve training convergence and yield slight gains on evaluation benchmarks, we observe a significant degradation in instruction-following performance. The resulting drop is substantial enough that the gains obtained in the earlier RLVR stage cannot be fully recovered.

C.5.3 Hyper-parameters

We use a batch size of 128, generating 16 rollout per prompt with a temperature of 1.0 and a top-p value of 1.0. We use a maximum response length of 16K during RLHF without applying overlong filtering. We adopt a learning rate of 3e-6 with AdamW (Kingma & Ba, 2014). We set the entropy loss coefficient to 0 and the KL loss coefficient to 0.03 to keep the model capabilities on other domains. The training takes around 30 steps.

C.6 Long-context RL

Following RLHF, we conduct a stage of long-context RL to further enhance the model's long-context understanding and reasoning capabilities. We use the NVIDIA Nano-v3 RL data blend (Blakeman

et al., 2025a), but restrict this phase to long-context datasets only. In our experiments, incorporating other domains during long-context RL negatively affects performance on unrelated benchmarks, motivating this domain-specific training setup.

We adopt the Nemo-Gym RL environment (NVIDIA, 2025c) and use Qwen3-235B-A22B-Instruct-2507 as an LLM judge to evaluate model rollouts for question answering tasks. During training, input sequences are limited to 32K tokens, and the maximum sequence length is set to 49K tokens without applying overlength filtering.

We train with a batch size of 128, generating 16 rollouts per prompt with temperature 1.0 and top-p 1.0. Optimization is performed using AdamW (Kingma & Ba, 2014) with a learning rate of 3×10^{-6}, while both the entropy and KL loss coefficients are set to zero. Training runs for approximately 30 steps, as we observe a rapid increase in generated tokens beyond that point.

C.7 Code RL

C.7.1 Data Curation

We construct our Code RL training set from the Nemotron-Cascade coding corpus (Wang et al., 2026), which contains coding prompts sourced from modern competitive programming platforms such as AtCoder, Codeforces, and AIZU with robust test cases for reward verification. To improve training efficiency and strengthen deep reasoning, we aggressively filter out prompts that GPT-OSS-120B solves correctly in all 8 of 8 rollouts, yielding a compact final set of only 3.5K samples. We find that high-difficulty prompts paired with strong test cases are critical for further boosting model performance.

C.7.2 Training Details

We conduct Code RL using a batch size of 128 and a learning rate of 3×10^{-6} with the AdamW optimizer. Compared to Nemotron-Cascade, we increase the maximum response length during RL to 118K tokens and the number of rollouts per sample to 16, enabling the policy to better capture sparse reward signals on extremely difficult problems that require long reasoning traces. We adopt the strict binary reward function to avoid potential reward hacking and keep the whole training to be fully on-policy for stability. To support the resulting verification throughput of $128 \times 16 = 2,048$ code executions per RL step, we deploy an asynchronous reward verification server that completes each batch in 427.2 seconds across 384 CPU cores.

C.8 Software Engineering Reinforcement Learning (SWE RL)

C.8.1 Agentless RL

Training Details and Hyperparameters. To enhance the models' code repair capability, we adopt the same data source as Wang et al. (2026) for agentless code repair reinforcement learning (RL) training. Since most instances do not provide executable Docker environments, we employ GPT-OSS-120B as a reward model to evaluate the quality of code repairs generated by our models. Following Wang et al. (2026), for each instance we construct prompts using both the golden localization and the top-5 retrieved localizations, and filter out relatively easy samples. We perform agentless SWE RL with a batch size of $128 \times 16 = 2{,}048$ (128 prompts with 16 rollouts per prompt), a maximum sequence length of 98,304, and a learning rate of 3×10^{-6} using the AdamW optimizer. We sample responses with temperature 1.0 and top-p 1.0. During training, we mask the loss for prompts for which none of the rollouts receives a reward greater than 0.5. We observe that these difficult prompts degrade the stability and effectiveness of agentless SWE RL training. Our agentless RL training typically converges within 40–50 steps.

Can Agentless RL Training Helps Agentic Tasks? Table 7 shows that agentless RL training not only improves model performance within the agentless framework but also enhances the models' ability to solve SWE tasks in agentic settings. Note that for Agentless Mini evaluation, we employ a code embedding model, NV-Embed-Code (Sohrabizadeh et al., 2025), to retrieve 5 candidate files whose code contents are semantically similar to the problem context. This result suggests that

Table 7: Effectiveness of Agentless RL on SWE-bench Verified.

Scaffold	Agentless Mini		OpenHands	
	avg@4	pass@4	avg@4	pass@4
Init.	41.9%	55.2%	49.8%	64.2%
after Agentless RL	44.3%	57.4%	50.8%	65.0%

improving models' code repair capability alone can generalize across different scaffolds, consistent with the observations from Yang et al. (2026).

C.8.2 Execution-based RL for Agentic SWE Scaffold

Modern software engineering agents rely on scaffolding frameworks that coordinate repository interaction, tool calling, code editing, and test execution. Training agents to operate effectively within these environments requires optimizing not only individual model outputs but the entire problem-solving trajectory. To address this, we apply Reinforcement Learning from Verifiable Rewards (RLVR) directly within agentic SWE scaffolds, enabling end-to-end optimization of the full agent workflow. Our training environments integrate established OpenHands frameworks (Wang et al., 2025a), which provide structured tool usage, repository interaction, and iterative patch generation.

We train agents using execution-based reinforcement learning in fully executable software environments, where each episode corresponds to resolving a software issue instance from benchmarks such as SWE-bench. The agent operates inside an instrumented repository that exposes tools for file inspection, search, code editing, and test execution. Candidate patches generated by the agent are executed within the environment, which returns verifiable signals from compilation results and unit test outcomes, enabling automatic reward computation without human annotation. Through the OpenHands scaffolding framework, the agent iteratively localizes defects, proposes patches, and validates them through test execution. Environment feedback—including compilation errors, failing tests, or successful test passes—provides deterministic rewards that directly reflect functional correctness.

Specifically, we conduct execution-based agentic reinforcement learning with a batch size of 1024, corresponding to 16 prompts with 64 rollouts per prompt. The maximum context length is set to 256k tokens, and the agent is allowed up to 200 interaction turns, providing a larger reasoning token budget during agentic coding problem solving. Training data is drawn from SWE-Gym (Pan* et al., 2025) and R2E-Subset (Jain et al., 2025). We generate 16 rollouts per instance using our intermediate model and evaluate them using the verification pipeline. Instances for which all rollouts pass verification (100% accuracy), indicating overly simple problems, are removed from the dataset. For instances where none of the rollouts pass verification (0% accuracy), indicating extremely difficult problems, we randomly discard 90% of such cases to reduce their proportion in the training data.

C.9 Training Compute and Additional Design Details

Training compute. Table 8 reports the compute used by each RL stage. All stages are trained on 128 NVIDIA H100 GPUs, and the RL pipeline uses about 13K GPU hours in total. Most of the MOPD compute is spent on long-reasoning math samples. Teacher scoring in MOPD requires only a forward pass over the already generated student tokens, with no teacher generation.

When to stop each stage. We stop each stage based on two signals: the trade-off between marginal gains on the target domain and regressions on other domains, and signs of entropy collapse. In Code RL, for example, token-level entropy decreases steadily during training; adding more diverse and difficult prompts while filtering out easy ones effectively mitigates this issue.

Data mixture. Within a multi-domain stage, we use a mixture proportional to the number of SFT trajectories in each domain. We did not tune these ratios further.

Table 8: Approximate compute per RL stage.

Stage	GPU hours	Tokens	FLOPs
IF-RL	4,000	9.5×10^8	4.0×10^{19}
Multi-domain RL	1,440	8.3×10^8	4.1×10^{19}
MOPD	2,000	2.61×10^9	4.7×10^{20}
RLHF	1,665	2.3×10^8	1.1×10^{19}
Long-context RL	634	1.9×10^9	7.9×10^{19}
Code RL	3,100	3.2×10^9	5.6×10^{20}

Data contamination. The competitive-programming data used for training has a cutoff of August 2024, before the 2025 competitions. Competitive-math proof data comes from sources separate from the IMO, and we apply strict filtering with n-gram overlap detection against evaluation benchmarks.

D International Mathematical Olympiad (IMO)

D.1 IMO 2025

In Table 3, we evaluate Nemotron-Cascade-2-30B-A3B on the IMO 2025 problem set using a self-improving test-time scaling framework (Shao et al., 2025), in which the model iteratively generates candidate solutions, verifies them, and refines them based on its own feedback. Each round generates $128 \times 64 = 8{,}192$ responses, and problems P1 to P5 use 2, 2, 4, 2, and 3 rounds, respectively. Without test-time scaling, a single sample per problem scores 5/42. Remarkably, despite its relatively modest 30B-A3B scale, the model successfully solves the first five problems. These results are particularly encouraging, as they suggest that strong olympiad-level mathematical reasoning can emerge from a comparatively compact model when paired with effective inference-time scaling. There remain several promising directions for improvement: expert review indicates that some proofs are longer than necessary, include superfluous intermediate steps or definitions, occasionally expose traces of intermediate reasoning, and sometimes contain minor typographical issues. For Problem 2, the model adopts an analytic solution strategy, similar to OpenAI's approach, rather than a more geometric approach such as that used by Gemini Deep Think (IMO Gold).

D.2 IMO-ProofBench

As shown in Table 4, Nemotron-Cascade-2-30B-A3B achieves 72.9 on IMO-ProofBench with generate-verify-refine test-time scaling, placing it within 8 points of DeepSeek-Math-V2-671B-A37B despite using $10\times$ fewer active parameters. It reaches 90+ on Basic split and surpasses QED-Nano-4B (LM-Provers et al., 2026) by 18 points overall. Re-evaluating the provided DeepSeek-Math-V2 proofs under our LLM-judge setup yields a score within 4 points of the reported human rating, suggesting that our protocol does not substantially overestimate performance (more details in Appendix F.1.2). In Figure 3, we show that increasing test-time compute improves Nemotron-Cascade-2-30B-A3B on IMO-ProofBench (Advanced), raising the score from 40.7 at round 1 to 53.4 at round 5 and narrowing the gap to DeepSeek-Math-V2 under the same grader.

E Competitive Coding

E.1 IOI 2025 and ICPC World Finals 2025

For IOI 2025, we adapt the IOI Test-Time Scaling pipeline from Nemotron-Cascade (Wang et al., 2026), which can be viewed as a multi-round *generate-select-submit* framework that exploits the model's reasoning ability under IOI's official rules. Each subtask is allotted at most 50 rounds. Within each round, we prompt our model to generate 40 candidate solutions, aggregated with (1) submission history with official judge verdicts from previous rounds, and (2) shared insights from high scored or fully solved subtasks within the same main task. The complete chat template is provided in Appendix I.1. Using this approach, we achieved full score on Problem 3 and 4, achieving a gold-medal score of 439.28 within at most $40 \times 50 = 2000$ model generations, while the score of

507.66 is achievable within 5000 generations. Notably, on Problem 2 which requires designing and optimizing a heuristic algorithm, our pipeline reached over 86 points in just 5 rounds (at most 200 model generations), demonstrating the effectiveness of self-refinement and cross-subtask insights.

For ICPC World Finals 2025, we generate up to 1000 solutions per problem and submit them for official evaluation after initial filtering. We successfully solved 10 out of 12 problems, achieving the #4 Gold medal placement, with 8 problems (except Problems A and I) solved within only 100 submissions.

E.2 Competitive Coding Benchmark Results

We evaluate our Nemotron-Cascade-2-30B-A3B model on various competitive coding benchmarks, including LiveCodeBench v6 (Jain et al., 2024), and LiveCodeBench Pro (Zheng et al., 2025)'s 25Q1 and 25Q2 splits. We also estimate Codeforces ELO score through simulated participation on **40** Div.1/Div.2 Codeforces Rounds held from 2501 to 2507. We report our avg@8 results under 128K-token thinking budget, the sampling temperature of 1.0 and the `top_p` of 0.95. For Tool-Integrated Reasoning (TIR) results, we allow our model to call a stateful Python executor for up to 100 calls. For baseline model evaluation, we follow their recommended inference configurations, ensuring a thinking budget of at least 128K tokens to at most 256K tokens. More evaluation details can be found in Appendix F and Appendix J.

As shown in Table 5, Nemotron-Cascade-2-30B-A3B achieves magnificent Pass@1 accuracy and ELO rating, even compared with frontier open-source models with over 100B total params, such as Nemotron-3-Super-120B-A12B, GPT-OSS-120B, and Qwen-3.5-122B-A10B. With Tool-Integrated Reasoning (TIR), our model's performance can be further boosted especially on hard problems, and match the strongest open-source models with more than 300B total parameters, such as Kimi-K2.5-1T-Thinking, Qwen-3.5-397B-A17B, and DeepSeek-v3.2-Speciale, which either lack TIR support for deep reasoning or perform poorly with Python TIR. Notably, Nemotron-Cascade-2-30B-A3B achieves above 0% on the LiveCodeBench Pro hard split within 8 attempts, demonstrating strong reasoning ability on problems that are extremely difficult even for humans.

F Benchmarks and Evaluation Setups

F.1 Math

F.1.1 Non-proof Math

For non-proof math reasoning tasks, we include

- **AIME 2025** [CC BY-NC-SA 4.0 via MathArena] (MAA, 2025) consists of 30 problems from American Invitational Mathematics Examination at 2025.
- **AIME 2026** [CC BY-NC-SA 4.0 via MathArena] (MAA, 2026) consists of 30 problems from American Invitational Mathematics Examination at 2026.
- **HMMT Feb 2025** [CC BY-NC-SA 4.0 via MathArena] (HMMT, 2025) consists of 30 problems from Harvard-MIT Mathematics Tournament 2025 February math competition.
- **IMO-AnswerBench** [CC BY 4.0] (Luong et al., 2025) consists of 400 problems with verifiable answers carefully chosen from past Olympiad competitions and then altered by experts to avoid memorization.

For Nemotron-Cascade-2-30B-A3 evaluated on AIME 2025, AIME 2026 and HMMT 2025 Feb, we set the thinking budget (maximum response length) to 131K tokens, the sampling temperature to 1.0, the top-p value to 1.0. For the with-tool setting, we enable tool use by appending a system-prompt postfix, allowing the model to call a stateful Python executor for up to 100 tool calls with a maximum response length of 131K tokens. For IMO-AnswerBench, we set to 256K tokens because we found the questions are significantly more difficult. We use and report the LLM-Judge score using GPT-OSS-120B (Agarwal et al., 2025) as the judge and the AnswerAutoGrader prompt (Luong et al., 2025) for answer correctness on IMO-AnswerBench as the short answers are complicated for rule-based verifier to compute. Following Liu et al. (2024b; 2026), we report avg@64 for AIME/HMMT and avg@16 for IMO-AnswerBench.

For baseline models, we use official numbers from their reports or evaluate them with the recommended settings if the official numbers are unavailable.

F.1.2 MATH PROOF

For math proof tasks, we include

- **IMO 2025** (IMO, 2025) consists of 6 problems from IMO 2025.
- **IMO-ProofBench** [CC BY 4.0] (Luong et al., 2025) is designed to evaluate the ability of AI models to construct comprehensive and valid mathematical arguments. This benchmark consists of 60 proof-based problems, curated to mirror the kinds of problems found in the IMO.

For Nemotron-Cascade-2-30B-A3, we apply test-time scaling following the DeepSeek-Math-V2 generate-verify-refine pipeline, using the same instructions. We implement this pipeline with NeMo-Skills (NVIDIA, 2025). We use the default hyperparameters from DeepSeek-Math-V2: 128 proof generations, 64 verifications per proof, selection of the top 32 proofs for refinement, and 8 verification analyses paired with each proof, prioritizing the lowest-rated analyses. We then generate 4 refined proofs and continue for up to 8 rounds, or until the average proof score reaches the threshold of 0.99999. We set the maximum generation length to 256K tokens, with temperature 1.0 and top-p 0.95.

For IMO-ProofBench Basic and 11 problems from the Advanced split (i.e., Problems 1, 4, 7, 13, 14, 17, 19, 22, 25, 26, and 28), we reduce the compute budget to 32 proof generations, 16 verifications, top 8 proofs, and 2 rounds to save compute. For IMO-ProofBench evaluation, we use DeepSeek-V3.2-Speciale to make sure the results are reproducible later and run 64 grading attempts with the ProofAutoGrader prompt (Luong et al., 2025). We found that reporting mean score yields 73.8 for DeepSeek-Math-V2 on the Advanced split, which is substantially more generous than the human rating of 61.9. We therefore adopt a simple aggregation rule based on analysis: `if any judge assigns a score of 0, the final score is set to 0; otherwise, return the mean score`. Under this rule, DeepSeek-Math-V2 obtains 57.7, which is much closer to the human rating and reduces the discrepancy from 11.9 points to 4.2 points.

F.2 CODE REASONING

For code generation tasks, we include

- **LiveCodeBench** [MIT] (Jain et al., 2024) contains diverse algorithm coding problems with unit tests, collected from AtCoder, LeetCode platforms. We evaluate models competitive coding capability on LiveCodeBench v6 (2024/08-2025/05, **454** problems in total). We report pass@1 accuracy in *thinking* mode, averaged over 8 generations (avg@8).
- **LiveCodeBench Pro** [Apache-2.0] (Zheng et al., 2025) contains daily-updated challenging competitive coding problems with strong unit tests, collected mainly from top-tier coding contests. We report pass@1 accuracy on Easy/Med difficulty splits in *thinking* mode, averaged over 8 generations (avg@8) on two recently released subsets: 2025Q1 (2025/01-2025/04, **166** problems in total) and 2025Q2 (2025/04-2025/07, **167** problems in total).
- **IOI and ICPC World Finals** represent the most challenging and prestigious annual algorithmic coding competitions, gathering the world's top human contestants. The IOI awards gold medals to approximately the top 8.3% (one-twelfth) of participants, while the ICPC World Finals (ICPCWF) limits gold medals to only the top 4 teams globally.
- **SciCode** [Apache-2.0] (Tian et al., 2024) serves as a challenging benchmark to evaluate model's ability on solving realistic scientific research tasks from STEM domains. It contains **338** subproblems from **80** main tasks.

For Nemotron-Cascade-2-30B-A3B evaluated on LiveCodeBench v6 and LiveCodeBench Pro, we use a 128K-token thinking budget, a sampling temperature of 1.0, a top-p of 0.95. For the with-tool setting, we enable tool use by appending a system-prompt postfix, allowing the model to call a stateful Python executor for up to 100 tool calls with a maximum response length of 131K tokens.

We evaluate baseline models with their recommended inference configurations, ensuring a thinking budget of at least 128K tokens.

F.3 KNOWLEDGE AND STEM

For knowledge reasoning tasks, we include:

- **MMLU-Redux** [CC BY 4.0] (Gema et al., 2024) is a benchmark consisting of a subset of 3,000 manually re-annotated questions across 30 MMLU subjects (Hendrycks et al., 2020), which eliminates the original annotation errors. We evaluate the models in *thinking* mode and, due to the large test set size, report exact match (EM) accuracy based on a single generation per question.
- **MMLU-Pro** [Apache-2.0] (Wang et al., 2024) is an enhanced version of the original MMLU benchmark that mitigates model saturation by expanding to over 12,000 graduate-level questions and increasing answer choices from four to ten. We report EM accuracy in *thinking* mode using one generation per question.
- **GPQA-Diamond** [CC BY 4.0] (Rein et al., 2024) is a benchmark for assessing an LLM's scientific reasoning capability. It consists of the highest quality 198 GPQA questions covering graduate-level physics, biology, and chemistry. We report pass@1 accuracy in *thinking* mode, averaged over 8 generations per question (avg@8) to reduce variance.
- **HLE** [MIT] (Phan et al., 2025) is a frontier academic reasoning benchmark spanning a broad range of expert-level subjects. We evaluate on its text-only split, which contains 2,158 examples.

For Nemotron-Cascade-2-30B-A3B evaluated on MMLU-Redux, MMLU-Pro, GPQA-Diamond and HLE in *thinking* mode, we use a temperature of 1.0, a top-p value of 0.95, and a 128K-token thinking budget (maximum response length). For HLE, we use the default system prompt and append "Please place your final answer inside `\boxed{}`" to each question, and use GPT-OSS-120B as the LLM judge for answer extraction and correctness verification with the prompt in Appendix I.2. Compared with the official HLE response format, which requests an explanation, an answer, and a confidence score, this boxed-answer prompt improves the accuracy by 6–7 points, primarily on the math subset, by better aligning with the answer format used in our math SFT data.

F.4 ALIGNMENT AND INSTRUCTION-FOLLOWING

For alignment tasks, we include:

- **ArenaHard 2.0** [Apache-2.0] (Li et al., 2024) is a human-preference alignment benchmark featuring 750 diverse and rigorous real-user prompts. The dataset is specifically structured with 500 prompts targeting open-ended software engineering problems and complex mathematical questions, while the remaining 250 focus on creative writing. It uses an automatic LLM-as-Judge approach to estimate human preferences relative to a baseline model, enabling fully automated, low-cost, and fast evaluation without human intervention. In our experiments, we report results without style control to allow for straightforward comparison with the officially reported numbers of other models. We evaluate the models in *thinking* mode, and use GPT-4.1 as the automated judge.
- **IFBench** [ODC-BY-1.0 data; Apache-2.0 code] (Pyatkin et al., 2025) extends IFEval (Zhou et al., 2023) by introducing 58 new, diverse, and challenging verifiable out-of-domain instruction constraints. It provides a separate constraint list to ensure no overlap between training and test constraints, enabling evaluation of an LLM's generalization ability. The test set contains 294 prompts. We report pass@1 accuracy in *thinking* mode, averaged over 8 generations (avg@8).
- **Scale AI Multi-Challenge** [CC BY 4.0] (Deshpande et al., 2025) is a benchmark designed to evaluate LLMs in multi-turn conversations with human users. It consists of four challenge categories: Instruction Retention, Inference Memory, Reliable Versioned Editing, and Self-Coherence. These tasks require models to simultaneously perform accurate instruction following, effective context management, and in-context reasoning. The test set contains 273 conversations in total. We report pass@1 accuracy in thinking mode, averaged across 10 generations (avg@10).

For Nemotron-Cascade models evaluated on IFEval in *non-thinking* mode, on IFBench and ArenaHard in *thinking* mode, we use a temperature of 0.6, a top-p value of 0.95, and a maximum response length of 32K tokens. For baseline models, we use officially reported results whenever available; if such results are absent, we evaluate them using their recommended inference configuration or the same settings as ours.

F.5 Long Context and Context Learning

For long context and context learning tasks, we include:

- **AA-LCR** [Apache-2.0] (Team, 2025) consists of 100 challenging text-based questions that require reasoning over multiple long, real-world documents, including company reports, government consultations, legal documents, and academic papers. Each sample contains a document set averaging approximately 100k tokens. The questions are designed such that answers cannot be directly retrieved from the documents and instead require reasoning across multiple sources of information. We report pass@1 accuracy in thinking mode, averaged over 16 generations (avg@16).
- **LongBench v2** [Apache-2.0] (Bai et al., 2025) contains 503 challenging multiple-choice questions with context lengths ranging from 8k to 2M words. The benchmark spans six task categories: single-document QA, multi-document QA, long in-context learning, long dialogue history understanding, code repository understanding, and long structured data understanding. The questions are designed to be difficult; even human experts equipped with document search tools may require substantial time to answer them correctly. We evaluate models in thinking mode and report pass@1 accuracy averaged over four generations (avg@4).
- **NIAH@1M (Ruler Subset)** [Apache-2.0] refers to the needle-in-a-haystack (NIAH) tasks from the RULER benchmark (Hsieh et al., 2024). The NIAH test (Kamradt, 2023) assesses an LLM's long-context ability to retrieve a specific piece of information (the "needle") embedded within long distractor text (the "haystack"). The RULER benchmark defines four variants of this task: Single NIAH, Multi-keys NIAH, Multi-values NIAH, and Multi-queries NIAH. Following Blakeman et al. (2025a), we evaluate 100 instances from each category using a 1M-token context setting. Models are evaluated in reasoning-off mode, and we report pass@1 accuracy from a single generation (avg@1).
- **CL-Bench** (Dou et al., 2026) evaluates an LLM's ability to learn from provided context and apply the acquired knowledge to solve tasks, a process referred to as context learning. The benchmark contains 1,899 test samples spanning 500 complex contexts and 31,607 verification rubrics, all developed by experienced domain experts. The knowledge required to complete these tasks largely falls outside what existing models typically learn during pre-training, requiring models to learn directly from the provided context. Models are evaluated in thinking mode, and we report pass@1 accuracy from a single generation (avg@1).

F.6 Agentic Tasks

For agentic tasks, we include:

- **BFCL v4** [Apache-2.0] (Patil et al., 2025) offers a comprehensive agentic evaluation framework for LLMs, covering tasks such as web search, memory reading and writing, and function invocation across multiple programming languages. We follow the official BFCL V4 evaluation protocol and report scores across a combination of Agentic, multi-turn, live, and non-live categories. Models are evaluated in thinking mode, and we report pass@1 accuracy based on a single generation (avg@1).
- **SWE-bench Verified** [MIT] (OpenAI, 2024) is a subset of the original test set from SWE-bench (Jimenez et al., 2023), consisting of 500 samples verified to be non-problematic by human annotators. We evaluate models in *non-thinking* mode and report pass@1 accuracy, averaged over 4 generations per prompt (avg@4).
- **τ^2-Bench** [MIT] (Barres et al., 2025) evaluates multi-turn customer-service agents in environments with explicit policies, tool use, and shared world-state updates. We evaluate on the three official subsets: airline (50 examples), retail (114 examples), and telecom (114 examples). To

keep the standard error within 1.5, we report avg@16 on airline and avg@8 on both retail and telecom.

- **Terminal Bench 2.0** [Apache-2.0](Merrill et al., 2026) is adopted for evaluating agents in terminal-based environments, which comprises of 89 human-validated tasks across specialized fields such as scientific computing, machine learning, and system administration. Moving beyond simple code generation, this benchmark focuses on end-to-end workflows, requiring agents to demonstrate proficiency in holistic operations like model training, system configuration, and software debugging rather than just producing isolated functions. We evaluate the model using the default Terminus-2 scaffolding. We report avg@5 task success rate.

For SWE-bench Verified, we use the OpenHands scaffold (Wang et al., 2025a) as the agentic coding evaluation framework. We adopt a full interaction retention policy for agent trajectories, preserving the complete history of tool calls, observations, and model outputs across turns. This includes prior file views, search results, executed commands, and intermediate patches, enabling the model to maintain state and reason effectively over long-horizon debugging processes. We set the maximum context length to 256K tokens and allow up to 200 turns, consistent with our execution-based agentic SWE-RL training configuration. Notably, this evaluation setup closely mirrors our training environment, as both rely on execution-based feedback and multi-turn interaction within the same tool-augmented scaffold. This alignment reduces train–test mismatch and enables the model to more effectively transfer learned behaviors, such as iterative debugging, hypothesis refinement, and tool-driven reasoning, to the evaluation setting.

For τ^2-Bench evaluation, we adopt a *latest-turn thought retention* policy for managing reasoning traces in multi-turn interactions: we retain the model's reasoning content after the most recent user turn, while discarding reasoning content from earlier turns. The official τ^2-Bench evaluation code follows a *no thought carry-over* policy, which removes all prior reasoning content; in our experiments, this evaluation setup consistently reduces scores by 3–5 points relative to latest-turn thought retention. We attribute this gap to train–test mismatch, since our SFT data for τ^2-style interactions is constructed with the same latest-turn thought retention policy, which is also the thought-state management strategy used in Nemotron-3-Nano-v3 and DeepSeek-V3.2. For the telecom subset, we additionally modify the system prompt to emphasize the dual-control setting by repeating the instruction "Make sure you guide the user through the steps, do not perform user-side actions yourself." three times. We also tested a *full thought retention* policy, which preserves reasoning content from all previous turns and more closely matches RL training, but found it gives similar accuracy to latest-turn thought retention while incurring substantially longer contexts. We therefore report our final τ^2-Bench results using latest-turn thought retention.

F.7 MULTILINGUAL

For multilingual tasks, we include:

- **MMLU-ProX** [MIT] (Xuan et al., 2025) expands the challenging MMLU-Pro benchmark to include 29 languages. Following Blakeman et al. (2025a), six languages are selected for evaluation: English (en), German (de), Spanish (es), French (fr), Italian (it), and Japanese (ja). The model is evaluated in thinking mode, and we report pass@1 accuracy from a single generation (avg@1).

- **WMT24++** [Apache-2.0] (Deutsch et al., 2025) extends the WMT24 machine translation benchmark to cover 55 languages. Following Blakeman et al. (2025a), we evaluate on five translation pairs: English to German (en $\rightarrow$ de), English to Spanish (en $\rightarrow$ es), English to French (en $\rightarrow$ fr), English to Italian (en $\rightarrow$ it), and English to Japanese (en $\rightarrow$ ja). We use XCOMET-XXL (Guerreiro et al., 2024) as the evaluation metric to assess the translation quality. Our model is evaluated in thinking mode, and we report pass@1 accuracy based on a single generation (avg@1).

G Training Data Summary

For anonymous review, we summarize the released training-data composition without listing public repository URLs. The final version will include permanent dataset URLs. The release-level licenses in Tables 9 and 10 are taken from the dataset cards; original source datasets retain their own terms.

Table 9: Summary of SFT training data with license prepared.

Domain	Source summary	# samples	License
Math	Non-proof prompts from Nemotron-Cascade-1-SFT and Nemotron-Math-v2, with responses generated by DeepSeek-V3.2, DeepSeek-V3.2-Speciale, and GPT-OSS-120B; proof prompts from Nemotron-Math-Proofs-v1.	5,226,364	Open Model License
Science	Prompts from Nemotron-Cascade-1-SFT and Nemotron-Science-v1, covering physics, chemistry, and biology; responses generated by GPT-OSS-120B.	2,717,163	Open Model License
General chat	Samples from Nemotron-Cascade-1-SFT and Nemotron-Instruction-Following-Chat-v1.	13,972,873	Open Model License
Instruction following	Samples from Nemotron-Cascade-1-SFT and Nemotron-Instruction-Following-Chat-v1.	820,263	Open Model License
Safety	Samples from Nemotron-SFT-Safety-v1.	3,570	Open Model License
Conversational agent	Prompts from Nemotron-Agentic-v1 and Nemotron-RL-Agentic-Conversational-Tool-Use-Pivot-v1; responses generated by Qwen3-235B-A22B-Thinking-2507, Qwen3-32B, Qwen3-235B-A22B-Instruct-2507, and GPT-OSS-120B.	822,213	Open Model License
Software engineering agent	Agentless samples from Nemotron-Cascade-1-SFT-SWE; agentic samples from SWE-Gym, SWE-rebench, and R2E-Gym-Subset.	439,610	Open Model License
Terminal agent	Samples from Nemotron-Terminal-Corpus.	822,213	Open Model License

H Training Hyperparameters

We list the training hyperparameters for the Nemotron-Cascade-2-30B-A3B during all stages in Table 11, 13, 14.

Table 10: Summary of RL training data with license prepared.

Subset	Source summary	# samples	License
IF-RL	Instruction-following RL data from Nemotron-RL-instruction_following, with formatting cleanup for selected keyword-argument fields.	45,879	ODC-BY-1.0
Multi-domain RL	Multi-choice question answering, workplace assistant, and structured-output instruction following; mixture weights are 55%, 30%, and 15%, respectively.	18,147	ODC-BY-1.0
MOPD	On-policy distillation instances from AceReason-Math (1,853), instruction following (1,854), workplace assistant (610), STEM (927), and structured outputs (927).	6,171	ODC-BY-1.0
SWE-RL	Software-engineering RL workflows from SWE-Gym (20%) and R2E-Gym-Subset (80%).	3,612	ODC-BY-1.0

Table 11: Training hyperparameters for Nemotron-Cascade-2-30B-A3B in SFT.

Hyperparameters	
Global batch size	64
Packed sequence length	256K
Max learning rate	5×10^{-5}
Min learning rate	5×10^{-6}
Learning rate warmup steps	200
Scheduler	cosine
Max Steps	40,000
Optimizer	AdamW
Optimizer config	$\beta_1 = 0.9, \beta_2 = 0.98$
Weight decay	0.1
# of training steps	33,000

Table 12: Training hyperparameters of Nemotron-Cascade-2-30B-A3B in Cascade RL (IF-RL, Multi-domain RL, MOPD).

Hyper-parameters	IF-RL	Multi-domain RL	MOPD
Max response length	49K	49K	98K
Batch size	128	128	128
# Rollout size	16	16	4
Learning rate	3×10^{-6}	3×10^{-6}	3×10^{-6}
Steps	180	70	52
Optimizer	AdamW $\beta_1 = 0.9$ $\beta_2 = 0.95$	Adam $\beta_1 = 0.9$ $\beta_2 = 0.95$	AdamW $\beta_1 = 0.9$ $\beta_2 = 0.95$
Temperature	1.0	1.0	1.0
Top-p	1.0	1.0	1.0
Overlong filtering	False	True	False

Table 13: Training hyperparameters of Nemotron-Cascade-2-30B-A3B in Cascade RL (RLHF, Long-context RL, Code RL).

Hyper-parameters	RLHF	Long-context RL	Code RL
Max response length	16K	49K	118K
Batch size	128	128	128
# Rollout size	16	16	16
Learning rate	3×10^{-6}	3×10^{-6}	3×10^{-6}
Steps	25	30	22
Optimizer	AdamW $\beta_1 = 0.9$ $\beta_2 = 0.95$	Adam $\beta_1 = 0.9$ $\beta_2 = 0.95$	AdamW $\beta_1 = 0.9$ $\beta_2 = 0.95$
Temperature	1.0	1.0	1.0
Top-p	1.0	1.0	0.95
Overlong filtering	True	True	True

Table 14: Training hyperparameters of Nemotron-Cascade-2-30B-A3B model in execution-based agentic SWE-RL.

Hyperparameters	
# prompts per step	16
# rollout	64
Temperature	0.8
Max sequence length	$256k$
Max turn	200
Max learning rate	3×10^{-6}
Min learning rate	0
Learning rate warmup steps	10

I PROMPT TEMPLATES

I.1 PROMPT TEMPLATES FOR TEST-TIME SCALING ON IOI 2025

Write Python code to solve the problem. Please place the solution code in the following format:
```python
# Your solution code here
```
{problem_statement}
Below you are provided the accepted correct solutions but with different input constraints. You may use them as a reference for your insights.
==========
Different Constraints (for reference only):
{subtask_constraints}
Accepted Code:
[CODE]
==========
Different Constraints (for reference only):
...
==========
From here, you are also given your submission history containing **incorrect** code and their corresponding official judgement verdicts as reference – Official judgement verdicts and problem statement/conditions are 100% reliable. You should make improvements from them if they could help:
==========
Incorrect Code
[CODE]
Judgement Verdict: [VERDICT], Score: [SCORE]
==========
Incorrect Code
...
==========

I.2 HLE JUDGE PROMPT

Judge whether the following [response] to [question] is correct or not based on the precise and unambiguous [correct_answer] below.

[question]: {question}

[response]: {response}

Your judgement must be in the format and criteria specified below:

extracted_final_answer: The final exact answer extracted from the [response]. Put the extracted answer as 'None' if there is no exact, final answer to extract from the response.

[correct_answer]: {correct_answer}

reasoning: Explain why the extracted_final_answer is correct or incorrect based on [correct_answer], focusing only on if there are meaningful differences between [correct_answer] and the extracted_final_answer. Do not comment on any background to the problem, do not attempt to solve the problem, do not argue for any answer different than [correct_answer], focus only on whether the answers match.

correct: Answer 'yes' if extracted_final_answer matches the [correct_answer] given above, or is within a small margin of error for numerical problems. Answer 'no' otherwise, i.e. if there if there is

any inconsistency, ambiguity, non-equivalency, or if the extracted answer is incorrect.

confidence: The extracted confidence score between 0|%| and 100|%| from [response]. Put 100 if there is no confidence score available.

J ELO RATING ANALYSIS

We perform ELO rating analysis on our Nemotron-Cascade-2-30B-A3B model based on **40** recent Div.1 and Div.2 Codeforces contests held between 2501–2507. Problems and evaluations are provided by LiveCodeBench Pro (Zheng et al., 2025). We adopt similar rating estimation approach as in Wang et al. (2026), by allowing model with up to $N = 8$ submissions to each contest problems, estimating model performance and relative ranking to human contestants with expected penalty consideration. We generate the model's responses using a temperature of 1.0, `top-p` of 0.95, and a maximum token budget of 128K. The performance details of our Nemotron-Cascade-2-30B-A3B model (**with** and **without** python-tool use) can be found in Table 15 and Table 16, respectively.

We observed our model's strong code reasoning ability on solving really tough problems and achieving high ranking even on some Div. 1 rounds (Round 999, 1012, 1015, 1021 etc.), while maintaining stable performance on solving easy-medium level problems. However, the models still has weakness on dealing with problems that requiring constructive algorithms, interactive manner, and hypothesis-driven ideas.

K BROADER IMPACTS

This work commits to release an open 30B MoE language model with strong reasoning, coding, and agentic capabilities. Potential positive impacts include improving access to capable open models, supporting reproducible research on RL post-training, and reducing experimentation costs through parameter-efficient inference. Potential risks include misuse for misleading content, unsafe or malicious code generation, biased outputs, and overreliance on model predictions in high-stakes settings. We recommend task-specific safety evaluation, appropriate deployment safeguards, and human oversight for sensitive applications.

Table 15: Nemotron-Cascade-2-30B-A3B performance details on 40 Div.1 and Div.2 Codeforces Rounds ranging from 2501 to 2507 **without** python-tool use. We attempt each problem with $N = 8$ times in total. For regular codeforces rounds, we present the score after considering expected penalties for each problem. For ICPC style rounds, we mark passed/failed problems as + and - correspondingly. We compute the estimated rank to human contestants and the corresponding Elo score as shown in rightmost two columns.

Contest Name	Contest Problems												Score	Penalty	Est. Rank	ELO
Hello 2025	A 500.00	B 1000.00	C 1493.75	D 2235.71	E1 0.0	E2 1900.00	F 0.0	G 3650.00	H 0.0				10779.46	-	13/16703	3449
Codeforces Round 996 (Div. 2)	A 500.00	B 993.75	C 1475.00	D 0.0	E 2825.00	F 0.0							5793.75	-	2/21232	2198
Codeforces Round 997 (Div. 2)	A 493.75	B 1250.00	C 1475.00	D 0.0	E 2225.00	F1 2710.00	F2 1225.00						9378.75	-	1/18823	2198
IAEPC Preliminary Contest (Codeforces Round 999, Div. 1 + Div. 2)	A 500.00	B 1000.00	C 1500.00	D 1493.75	E 1960.00	F1 0.0	F2 0.0	G 0.0	H1 2825.00	H2 0.0	I 0.0		9278.75	-	43/12647	3076
Codeforces Round 1000 (Div. 2)	A 500.00	B 985.71	C 1500.00	D 2250.00	E 2687.50	F1 1687.50	F2 1325.00						10935.71	-	1/17169	2200
Ethflow Round 1 (Codeforces Round 1001, Div. 1 + Div. 2)	A 500.00	B 993.75	C 1000.00	D 0.0	E1 0.0	E2 0.0	F 0.0	G 0.0	H 0.0				2493.75	-	1727/16234	1898
Codeforces Round 1002 (Div. 2)	A 500.00	B 975.00	C 0.0	D 1825.00	E1 0.0	E2 0.0							3300.00	-	1102/19443	1882
Codeforces Round 1004 (Div. 1)	A 0.0	B 687.50	C 1250.00	D1 743.75	D2 0.0	E 0.0	F 0.0						2681.25	-	145/1030	2666
Codeforces Round 1004 (Div. 2)	A 500.00	B 960.00	C 0.0	D 0.0	E 1687.50	F 2250.00	G 0.0						5397.50	-	8/16749	2098
Codeforces Round 1005 (Div. 2)	A 493.75	B 1000.00	C 1243.75	D 1735.71	E 2075.00	F 2650.00							9198.21	-	1/17621	2260
Educational Codeforces Round 174 (Rated for Div. 2)	A +	B +	C +	D +	E -	F -							4	2.86	156/16701	2242
Educational Codeforces Round 175 (Rated for Div. 2)	A +	B +	C +	D +	E -	F -							4	0.00	234/16060	2195
Codeforces Round 1007 (Div. 2)	A 500.00	B 1000.00	C 1485.71	D1 1743.75	D2 1225.00	E 2475.00	F 0.0						8429.46	-	1/16254	2198
Codeforces Round 1008 (Div. 1)	A 500.00	B 0.0	C 1500.00	D 0.0	E 0.0	F 0.0	G 0.0						2000.00	-	355/909	2312
Codeforces Round 1008 (Div. 2)	A 500.00	B 750.00	C 1250.00	D 1575.00	E 0.0	F 2750.00	G 0.0						6825.00	-	9/14641	2008
Educational Codeforces Round 176 (Rated for Div. 2)	A +	B +	C +	D +	E +	F -							5	10.86	2/18159	2198
Codeforces Round 1011 (Div. 2)	A 500.00	B 1250.00	C 1250.00	D 1743.75	E 2500.00	F1 1993.75	F2 900.00						10137.50	-	1/15906	2200
Codeforces Round 1012 (Div. 1)	A 710.00	B1 975.00	B2 325.00	C1 975.00	C2 0.0	D 0.0	E 0.0						2985.00	-	24/653	3057
Codeforces Round 1012 (Div. 2)	A 500.00	B 960.00	C 1750.00	D 1960.00	E1 1975.00	E2 825.00	F1 1975.00	F2 0.0					9945.00	-	1/8536	2007
Codeforces Round 1014 (Div. 2)	A 500.00	B 750.00	C 1250.00	D 1750.00	E 2250.00	F 0.0							6500.00	-	2/15842	2213
Teza Round 1 (Codeforces Round 1015, Div. 1 + Div. 2)	A 750.00	B 1000.00	C 1500.00	D 1735.71	E 2235.71	F 2825.00	G1 2475.00	G2 0.0	H 0.0				12521.43	-	4/11206	3830
Neowise Labs Contest 1 (Codeforces Round 1018, Div. 1 + Div. 2)	A 500.00	B 750.00	C 1500.00	D 1650.00	E 0.0	F 0.0	G 0.0	H 0.0					4400.00	-	493/12771	2312
Codeforces Round 1019 (Div. 2)	A 500.00	B 1000.00	C 1500.00	D 1825.00	E 0.0	F 0.0							4825.00	-	47/14465	2202
Codeforces Round 1021 (Div. 1)	A 493.75	B 900.00	C 0.0	D 1825.00	E 0.0	F 0.0							3218.75	-	75/651	2760
Codeforces Round 1021 (Div. 2)	A 500.00	B 1250.00	C 1493.75	D 2150.00	E 0.0	F 3075.00							8468.75	-	1/5824	2019
Educational Codeforces Round 178 (Rated for Div. 2)	A +	B +	C +	D +	E +	F +	G -						6	12.50	4/11706	2215
Codeforces Round 1022 (Div. 2)	A 500.00	B 1187.50	C 1400.00	D 0.0	E 0.0	F 0.0							3087.50	-	308/11127	2132
Codeforces Round 1023 (Div. 2)	A 250.00	B 750.00	C 1493.75	D 1937.50	E 0.0	F1 2075.00	F2 0.0						6506.25	-	6/11636	2209
Codeforces Round 1024 (Div. 1)	A 485.71	B 1243.75	C 0.0	D 0.0	E 0.0	F 0.0							1729.46	-	477/857	2149
Codeforces Round 1024 (Div. 2)	A 250.00	B 500.00	C 985.71	D 1743.75	E 0.0	F 0.0							3479.46	-	34/11201	1998
Codeforces Round 1025 (Div. 2)	A 500.00	B 985.71	C1 1243.75	C2 575.00	C3 500.00	D 1687.50	E 2493.75	F 0.0					7985.71	-	1/15945	2197
Codeforces Round 1026 (Div. 2)	A 500.00	B 750.00	C 1500.00	D 1960.00	E 2250.00	F 2937.50							9897.50	-	1/17668	2198
Codeforces Round 1028 (Div. 1)	A 500.00	B 0.0	C 0.0	D 2210.00	E 0.0	F1 0.0	F2 0.0						2710.00	-	75/956	2865
Codeforces Round 1028 (Div. 2)	A 493.75	B 750.00	C 1250.00	D 0.0	E 0.0	F 2960.00							5453.75	-	4/18314	2018
Educational Codeforces Round 179 (Rated for Div. 2)	A +	B -	C +	D +	E +	F +	G -						5	60.00	94/12301	2231
Codeforces Round 1030 (Div. 2)	A 500.00	B 975.00	C 1000.00	D1 1243.75	D2 960.00	E 2325.00	F 0.0						7003.75	-	2/18335	2205
Codeforces Round 1031 (Div. 2)	A 500.00	B 735.71	C 0.0	D 0.0	E 0.0	F 2825.00							4060.71	-	20/11032	2216
Codeforces Round 1033 (Div. 2) and CodeNite 2025	A 493.75	B 750.00	C 1250.00	D 1735.71	E 2493.75	F 2900.00	G 0.0						9623.21	-	1/12948	2216
Educational Codeforces Round 180 (Rated for Div. 2)	A +	B +	C +	D +	E +	F -							5	33.75	8/17128	2253
Codeforces Round 1035 (Div. 2)	A 500.00	B 1000.00	C 1485.71	D 0.0	E 0.0	F 0.0							2985.71	-	587/15624	2008

Table 16: Nemotron-Cascade-2-30B-A3B performance details on 40 Div.1 and Div.2 Codeforces Rounds ranging from 2501 to 2507 **with** python-tool use. We attempt each problem with $N = 8$ times in total. For regular codeforces rounds, we present the score after considering expected penalties for each problem. For ICPC style rounds, we mark passed/failed problems as + and - correspondingly. We compute the estimated rank to human contestants and the corresponding Elo score as shown in rightmost two columns.

Contest Name	Contest Problems											Score	Penalty	Est. Rank	ELO
Hello 2025	A	B	C	D	E1	E2	F	G	H			11712.50	-	11/16703	3497
	500.00	1000.00	1500.00	2225.00	937.50	1900.00	0.0	3650.00	0.0						
Codeforces Round 996 (Div. 2)	A	B	C	D	E	F						5025.00	-	2/21232	2198
	500.00	975.00	1475.00	2075.00	0.0	0.0									
Codeforces Round 997 (Div. 2)	A	B	C	D	E	F1	F2					11187.50	-	1/18823	2198
	493.75	1250.00	1500.00	1900.00	2075.00	2743.75	1225.00								
IAEPC Preliminary Contest (Codeforces Round 999, Div. 1 + Div. 2)	A	B	C	D	E	F1	F2	G	H1	H2	I	9416.96	-	40/12647	3097
	500.00	1000.00	1493.75	1485.71	2000.00	0.0	0.0	0.0	2937.50	0.0	0.0				
Codeforces Round 1000 (Div. 2)	A	B	C	D	E	F1	F2					10981.25	-	1/17169	2200
	500.00	1000.00	1500.00	2243.75	2725.00	1687.50	1325.00								
Ethflow Round 1 (Codeforces Round 1001, Div. 1 + Div. 2)	A	B	C	D	E1	E2	F	G	H			2493.75	-	1727/16234	1898
	500.00	993.75	1000.00	0.0	0.0	0.0	0.0	0.0	0.0						
Codeforces Round 1002 (Div. 2)	A	B	C	D	E1	E2						3300.00	-	1102/19443	1882
	500.00	975.00	0.0	1825.00	0.0	0.0									
Codeforces Round 1004 (Div. 1)	A	B	C	D1	D2	E	F					2743.75	-	122/1030	2721
	0.0	743.75	1250.00	750.00	0.0	0.0	0.0								
Codeforces Round 1004 (Div. 2)	A	B	C	D	E	F	G					5487.50	-	6/16749	2098
	500.00	993.75	0.0	0.0	1743.75	2250.00	0.0								
Codeforces Round 1005 (Div. 2)	A	B	C	D	E	F						9230.71	-	1/17621	2260
	500.00	985.71	1250.00	1710.00	2210.00	2575.00									
Educational Codeforces Round 174 (Rated for Div. 2)	A	B	C	D	E	F						4	2.50	156/16701	2242
	+	+	+	+	-	-									
Educational Codeforces Round 175 (Rated for Div. 2)	A	B	C	D	E	F						5	5.00	3/16060	2198
	+	+	+	+	+	-									
Codeforces Round 1007 (Div. 2)	A	B	C	D1	D2	E	F					8443.75	-	1/16254	2198
	500.00	1000.00	1500.00	1725.00	1225.00	2493.75	0.0								
Codeforces Round 1008 (Div. 1)	A	B	C	D	E	F	G					2000.00	-	355/909	2312
	500.00	0.0	1500.00	0.0	0.0	0.0	0.0								
Codeforces Round 1008 (Div. 2)	A	B	C	D	E	F	G					6975.00	-	5/14641	2008
	500.00	750.00	1250.00	1725.00	0.0	2750.00	0.0								
Educational Codeforces Round 176 (Rated for Div. 2)	A	B	C	D	E	F						5	2.50	2/18159	2198
	+	+	+	+	+	-									
Codeforces Round 1011 (Div. 2)	A	B	C	D	E	F1	F2					10137.50	-	1/15906	2200
	500.00	1250.00	1250.00	1743.75	2493.75	2000.00	900.00								
Codeforces Round 1012 (Div. 1)	A	B1	B2	C1	C2	D	E					2693.75	-	66/653	2745
	725.00	975.00	0.0	993.75	0.0	0.0	0.0								
Codeforces Round 1012 (Div. 2)	A	B	C	D	E1	E2	F1	F2				9193.75	-	1/8536	2007
	500.00	1000.00	1750.00	1975.00	1975.00	0.0	1993.75	0.0							
Codeforces Round 1014 (Div. 2)	A	B	C	D	E	F						6500.00	-	2/15842	2213
	500.00	750.00	1250.00	1750.00	2250.00	0.0									
Teza Round 1 (Codeforces Round 1015, Div. 1 + Div. 2)	A	B	C	D	E	F	G1	G2	H			9723.21	-	55/11206	3008
	750.00	1000.00	1500.00	1743.75	2243.75	0.0	2485.71	0.0	0.0						
Neowise Labs Contest 1 (Codeforces Round 1018, Div. 1 + Div. 2)	A	B	C	D	E	F	G	H				6397.50	-	70/12771	2933
	500.00	750.00	1500.00	1687.50	1960.00	0.0	0.0	0.0							
Codeforces Round 1019 (Div. 2)	A	B	C	D	E	F						7725.00	-	2/14465	2202
	500.00	1000.00	1500.00	1825.00	0.0	2900.00									
Codeforces Round 1021 (Div. 1)	A	B	C	D	E	F						4899.46	-	21/651	3143
	493.75	985.71	1460.00	1960.00	0.0	0.0									
Codeforces Round 1021 (Div. 2)	A	B	C	D	E	F						11399.46	-	1/5824	2019
	500.00	1250.00	1493.75	2235.71	2710.00	3210.00									
Educational Codeforces Round 178 (Rated for Div. 2)	A	B	C	D	E	F	G					6	12.11	4/11706	2215
	+	+	+	+	+	+	-								
Codeforces Round 1022 (Div. 2)	A	B	C	D	E	F						3235.71	-	300/11127	2137
	500.00	1250.00	1485.71	0.0	0.0	0.0									
Codeforces Round 1023 (Div. 2)	A	B	C	D	E	F1	F2					6537.50	-	6/11636	2209
	250.00	743.75	1493.75	1900.00	0.0	2150.00	0.0								
Codeforces Round 1024 (Div. 1)	A	B	C	D	E	F						4075.00	-	156/857	2590
	500.00	1250.00	0.0	2325.00	0.0	0.0									
Codeforces Round 1024 (Div. 2)	A	B	C	D	E	F						6325.00	-	2/11201	1998
	250.00	500.00	1000.00	1750.00	0.0	2825.00									
Codeforces Round 1025 (Div. 2)	A	B	C1	C2	C3	D	E	F				8087.14	-	1/15945	2197
	500.00	985.71	1235.71	710.00	460.00	1710.00	2485.71	0.0							
Codeforces Round 1026 (Div. 2)	A	B	C	D	E	F						7931.25	-	8/17668	2198
	500.00	750.00	1500.00	0.0	2243.75	2937.50									
Codeforces Round 1028 (Div. 1)	A	B	C	D	E	F1	F2					3960.71	-	36/956	3093
	500.00	1225.00	0.0	2235.71	0.0	0.0	0.0								
Codeforces Round 1028 (Div. 2)	A	B	C	D	E	F						7454.46	-	1/18314	2018
	493.75	750.00	1250.00	1975.00	0.0	2985.71									
Educational Codeforces Round 179 (Rated for Div. 2)	A	B	C	D	E	F	G					6	40.11	3/12301	2231
	+	+	+	+	+	+	-								
Codeforces Round 1030 (Div. 2)	A	B	C	D1	D2	E	F					7160.71	-	2/18335	2205
	500.00	993.75	1000.00	1243.75	937.50	2485.71	0.0								
Codeforces Round 1031 (Div. 2)	A	B	C	D	E	F						4143.75	-	20/11032	2216
	500.00	743.75	0.0	0.0	0.0	2900.00									
Codeforces Round 1033 (Div. 2) and CodeNite 2025	A	B	C	D	E	F	G					9650.00	-	1/12948	2216
	500.00	750.00	1250.00	1750.00	2500.00	2900.00	0.0								
Educational Codeforces Round 180 (Rated for Div. 2)	A	B	C	D	E	F						5	10.86	8/17128	2253
	+	+	+	+	+	-									