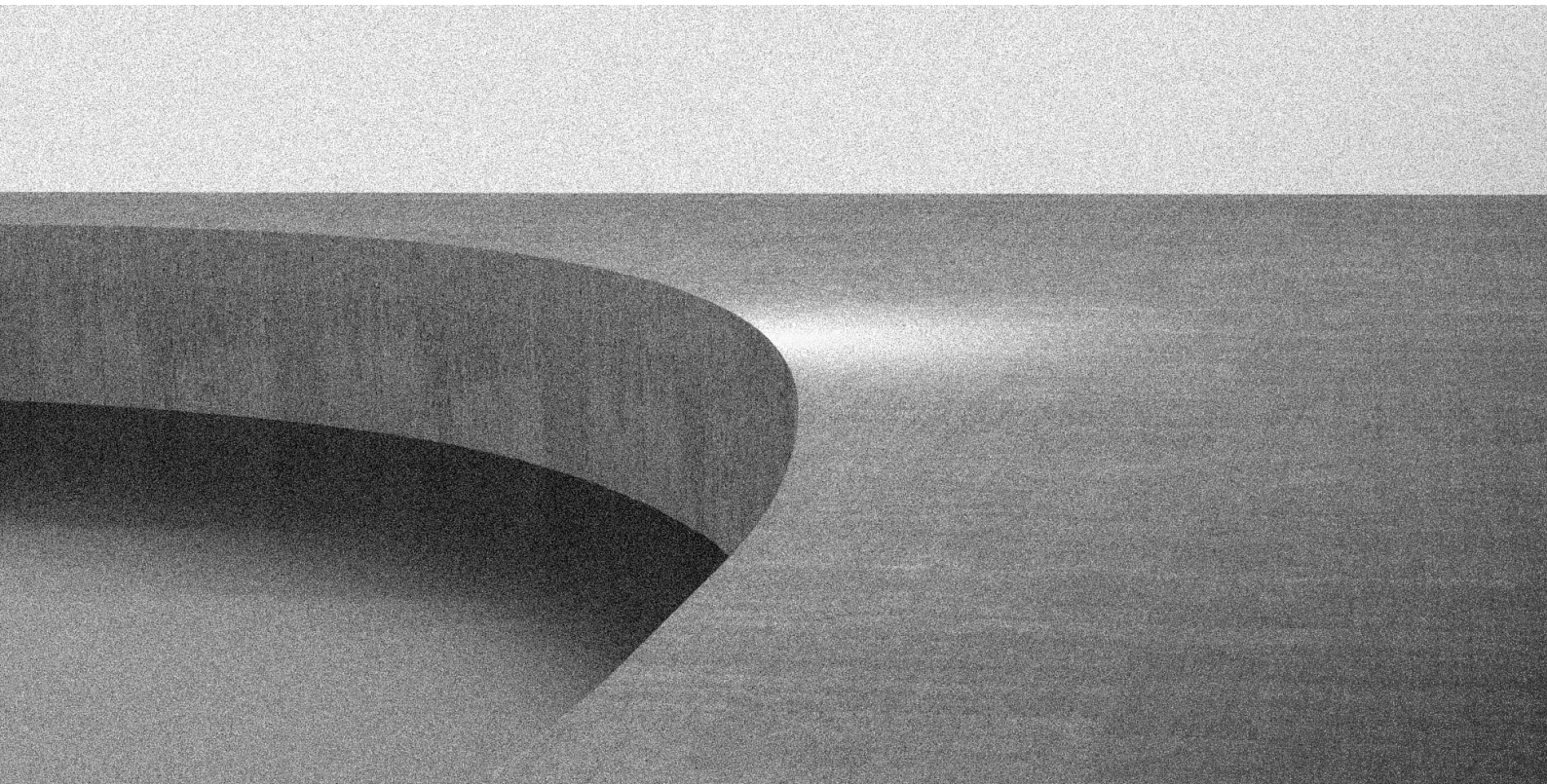


Limite 1B - Violetto

Technical Report

MARIO PRIGNANO, GABRIELE CIRILLO, ALESSIO MOROSINI, LUCA CERVAZ, ALESSANDRO BARTOLOCCI
EMANUELE RODOLÀ, GIULIO STARACE, FRANCESCO PAPPONE



Limite 1B - Violetto: Technical Report

Mario Prignano, Gabriele Cirillo, Alessio Morosini, Luca Cerovaz, Alessandro Bartolucci
Emanuele Rodolà, Giulio Starace, Francesco Pappone

Paradigma

ABSTRACT

We present Limite 1B - Violetto, a one-billion-parameter model designed for high-throughput mathematical reasoning. Limite is trained from scratch on 237 billion tokens from a carefully curated, mathematics-centric corpus and supports context lengths of up to 131k tokens. Its architecture and training recipe were developed together to maximize reasoning capability while keeping training and inference costs low. Post-training combines knowledge distillation, targeted supervised fine-tuning, and RLVR on a curated curriculum. Limite achieves 94.01% on AIME 2026 and 74.25% on BeyondAIME [1, 2], placing it among models many times its size on our evaluation suite. The model is deliberately only lightly instruction-tuned: it is intended primarily as a single-turn mathematical solver and as a fast, parallel component of larger systems, rather than as a general-purpose assistant. We release the model, its base and value checkpoints, and an optimized serving implementation under the Apache 2.0 license.

GitHub: <https://github.com/paradigma-inc/limite-violetto>

Hugging Face: <https://huggingface.co/paradigma-inc/models>

Blog: <https://paradigma.inc/blog/limite-1b-violetto/>

Correspondence: research@paradigma.inc

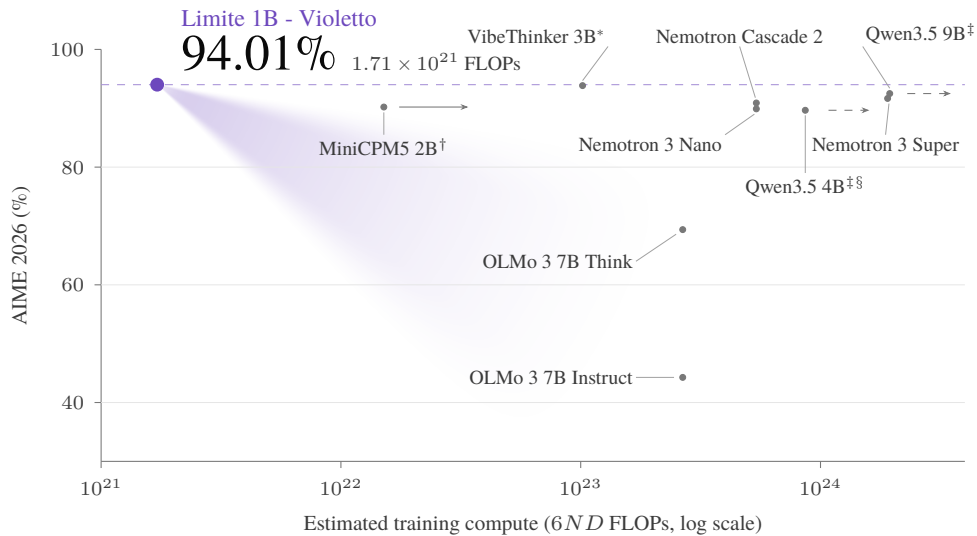


Figure 1: AIME 2026 accuracy against estimated training compute. Limite 1B - Violetto reaches 94.01% at 1.71×10^{21} FLOPs, one to three orders of magnitude below the models that match it. Appendix F explains the compute estimates, the score sources and the marked points.

1 INTRODUCTION

Progress in mathematical reasoning has often come with increases in model size, training compute, and test-time sampling. While this has produced increasingly capable systems, it also makes repeated sampling, long-context inference, and large-scale deployment expensive. Many applications require a different operating point: a model that can solve difficult problems while remaining small enough to run at relatively high throughput and to generate many candidate solutions in parallel. This motivates a complementary question to conventional scaling: how much mathematical capability can we concentrate into a model with approximately one billion parameters?

We introduce **Limite 1B - Violetto**, a dense autoregressive model built around this objective. Limite is trained from scratch for mathematics rather than adapted from a general-purpose base model. With one billion parameters, it supports contexts of up to 131k tokens, and is designed to provide high reasoning capability per parameter. The model is only lightly instruction-tuned and is intended primarily as a single-turn mathematical solver, as opposed to a general conversational assistant.

Efficiency guides the design of the entire training pipeline. Limite uses a deep architecture with a mixture of local and global attention, allowing it to perform many sequential transformations while limiting the cost of long-context inference. The model is pre-trained on 237 billion tokens drawn from a larger mathematics-centric corpus containing problem-solution pairs, synthetic mathematical material, and carefully filtered STEM-adjacent web text. The resulting base model already exhibits substantial mathematical ability before post-training.

Post-training is organized as a sequence of complementary stages. We begin with knowledge distillation from millions of filtered mathematical solutions, then alternate targeted supervised fine-tuning with RL. Rather than repeatedly training on broad, static datasets, each stage focuses on areas where the current checkpoint remains weak. Later RL stages also use a learned value model to improve credit assignment over long reasoning traces.

We evaluate Limite across seven benchmarks covering recent mathematical competitions and research-level problems. It obtains 94.01% on AIME 2026, 83.62% on HMMT February 2026, and 74.25% on BeyondAIME [1, 3, 2]. A more comprehensive evaluation is available in Section 5. These results are competitive with models ranging from several billion to tens of billions of parameters.

We envision Limite’s small size as possibly enabling a broader mode of use, where a controller may delegate mathematical questions to multiple Limite instances, inspect their outputs asynchronously, and combine promising lines of reasoning. This separates high-level coordination from high-frequency mathematical exploration and makes it possible to spend additional test-time compute through parallelism rather than through a single large model. The same specialization that makes Limite effective in this setting is also its principal limitation: outside mathematics, it may misinterpret a request or replace it with a superficially related mathematical task, an example of this is shown in Appendix H. Also within mathematics, Limite works best at mathematical reasoning, while it may struggle with knowledge of the literature or formalization and structured theorem proving.

We show that strong mathematical reasoning can be achieved with a restricted parameter count and relatively few steps of training. By jointly designing the architecture, data, optimization, and post-training process around a narrow objective, Limite concentrates substantial reasoning capability into a compact model that can be sampled from and deployed efficiently. We release the final model, its base and value checkpoints, our evaluation suite, and the serving implementation all under the Apache 2.0 license.

2 MODEL ARCHITECTURE

We designed Limite 1B to be as sample efficient as possible in its learning, without materially reducing training throughput. We evaluated architectural variants at shorter token horizons and selected a configuration that remained stable and efficient with our training settings.

Many architectural modifications are motivated primarily by converged model quality or inference efficiency, with less attention paid to wall-clock training time. We nevertheless still make some tweaks to reduce wall-clock overhead, inspired by modded nanogpt speedruns. Most additions are therefore applied only to selected layers or restricted portions of the residual stream, making them sparse in depth, width, or both. This keeps training throughput close to that of an otherwise standard modern Transformer. We detail our architecture in the following subsections and present a visual overview in Figure 2.

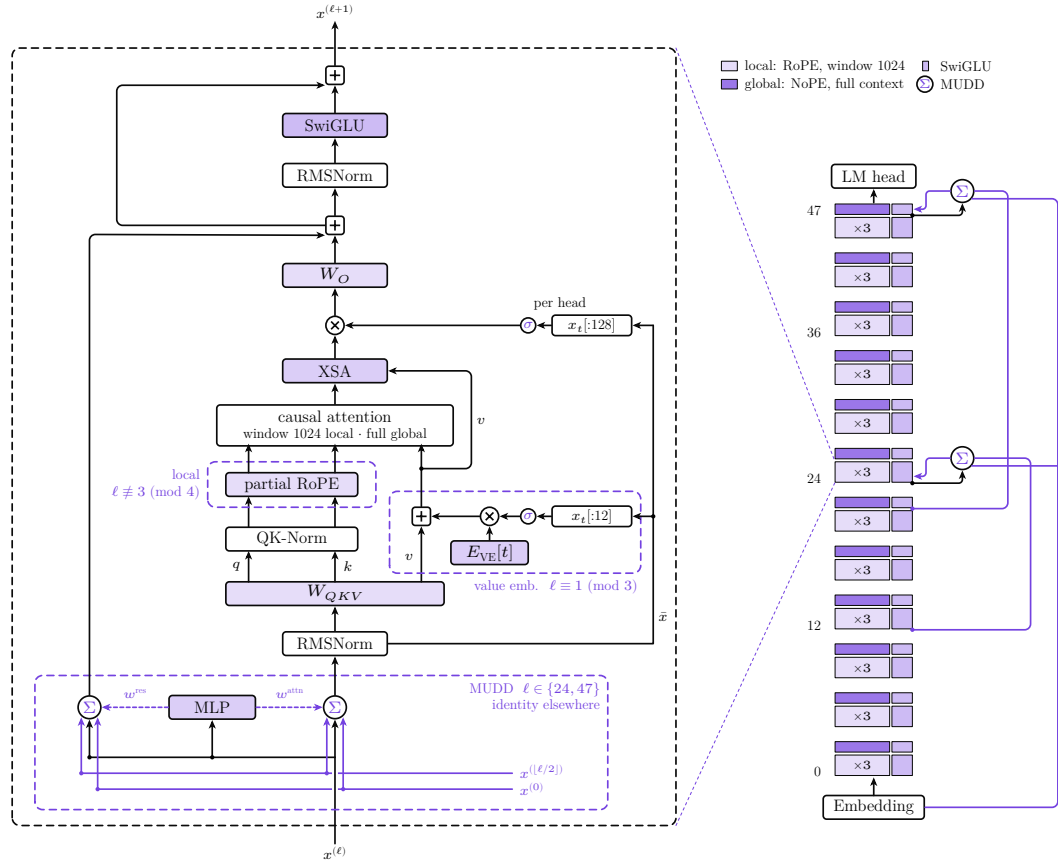


Figure 2: Architecture of Limite 1B - Violetto. Left: a Transformer block ℓ , showing the simplified MUDD mixers at its input, gated value embeddings on the value branch, and Partial RoPE on local attention [4, 5]. Right: the 48-layer stack, arranged as 12 repetitions of three local-attention layers followed by one global-attention layer. Value embeddings are applied every third layer ($\ell \equiv 1 \pmod{3}$); they are omitted from the stack view for clarity.

2.1 BACKBONE AND BLOCK STRUCTURE

Limite 1B is a 48-layer RMS-pre-norm Transformer [6, 7, 8] with a hidden dimension of $d = 1280$ and tied embeddings [9]. It uses a repeating pattern of three local-attention [10, 11, 12] layers followed by one global-attention layer [13, 14]. The local layers have a window size of 1024 and use

Partial RoPE [15, 16, 17] with a base of 1024, whereas the global layers use NoPE [5, 4]. This hybrid pattern reduces KV-cache requirements while retaining regular access to the full context.

The model is considerably deeper than other Transformers of a similar size. This choice reflects our hypothesis that difficult mathematical reasoning benefits from a larger number of sequential operations, represented here through network depth [18, 19].

Logits soft-capping Finally, to make training more stable, the output logits are soft-capped with a shifted sigmoid

$$\tilde{z} = 23 \sigma((z + 5)/7.5)$$

where z is produced by the tied unembedding projection applied to the output of the last layer, a variant from the modded-nanogpt speedrun [20] of the tanh logit soft-capping of Gemma 2 [13].

2.2 MLP

The feed-forward layer is a standard bias-free SwiGLU [21, 22] MLP without activation clamping. Given a residual state $x_t \in \mathbb{R}^d$ and its normalized representation

$$\bar{x}_t = \text{RMSNorm}(x_t),$$

the MLP output is

$$m_t = W_d (\text{SiLU}(W_g \bar{x}_t) \odot W_u \bar{x}_t),$$

where

$$W_g, W_u \in \mathbb{R}^{d_{\text{ff}} \times d}, \quad W_d \in \mathbb{R}^{d \times d_{\text{ff}}}, \quad d_{\text{ff}} = 3328.$$

The output is added to the residual stream using learned per-layer scalars:

$$x_t \leftarrow \lambda_{\text{mlp}} x_t + \lambda'_{\text{mlp}} m_t.$$

2.3 ATTENTION

Each attention layer uses gated Grouped-Query Attention [23] with 10 query heads and 2 key-value (KV) heads. We augment the standard attention computation with gated value embeddings [20], Exclusive Self-Attention (XSA) [24] and gated attention [25].

Let $x_t \in \mathbb{R}^d$ denote the residual state at position t , and let

$$\bar{x}_t = \text{RMSNorm}(x_t)$$

be the normalized input to the attention layer. For clarity, we describe the computation for a single head, with head dimension $d_h = 128$. Queries, keys, and values are obtained as

$$q_t = W_q \bar{x}_t, \quad k_t = W_k \bar{x}_t, \quad v_t = W_v \bar{x}_t.$$

The queries and keys are then normalized as in Henry et al. [26] and positionally encoded:

$$\hat{q}_t = \text{PE}_\ell(\text{RMSNorm}(q_t)), \quad \hat{k}_t = \text{PE}_\ell(\text{RMSNorm}(k_t)),$$

where PE_ℓ denotes Partial RoPE in local layers and the identity map in global layers.

Gated value embeddings Every third layer, specifically those satisfying $\ell \equiv 1 \pmod{3}$, augments the projected value with a learned embedding of the input token [20]:

$$\hat{v}_t = v_t + \alpha_t \odot \mathbf{V}_t.$$

Here, $\mathbf{V}_t \in \mathbb{R}^{d_h}$ is retrieved from a learned lookup table

$$\mathcal{V} \in \mathbb{R}^{N_{\text{vocab}} \times d_h},$$

using the token at position t . Its contribution is controlled by a per-token, per-head gate,

$$\alpha_t = 2\sigma(W_{\text{gate}} \bar{x}_{t,1:12}),$$

which depends only on the first 12 channels of the normalized input [20]. Layers without value embeddings use $\hat{v}_t = v_t$.

Standard causal attention is then applied to $\hat{q}_t$, $\hat{k}_t$, and $\hat{v}_t$, producing an output z_t for each head.

Exclusive Self-Attention The attention output is subsequently processed by Exclusive Self-Attention [24], which removes a learned fraction of its component aligned with the corresponding value vector, and later gated, effectively making our layer a form of gated attention [25], in practice:

$$y_t = \beta_t \left(z_t - \tanh(\gamma) \frac{z_t^\top \hat{v}_t}{\|\hat{v}_t\|_2^2} \hat{v}_t \right).$$

The coefficient γ is a learned scalar for each head. The output is also modulated by a per-token, per-head gate,

$$\beta_t = 2\sigma(W_\beta \bar{x}_{t,1:128}),$$

which reads only the first 128 channels of the normalized residual stream.

2.4 SPARSE RESIDUAL MIXING

Modern Transformer architectures often modify the residual stream [27] to increase communication between distant layers [28, 29, 30]. Because a dense implementation would impose substantial training overhead, Limite uses a simplified MUDD-style residual mixer [31] at two locations in the network, effectively becoming an input-dependent form of a DenseFormer [32].

At layers $\ell \in \{24, 47\}$, the input to the Transformer block is replaced by a token-dependent linear combination of three residual states: the initial embedded representation, a state near the midpoint of the preceding stack, and the current layer input.

Let $x_t^{(\ell)}$ denote the residual state entering layer ℓ , with

$$x_t^{(0)} = \text{RMSNorm}(\mathbf{E}_t),$$

where $\mathbf{E}_t \in \mathbb{R}^d$ is the token embedding at position t . At each mixing site, we define the set of residual taps

$$\mathcal{T}_\ell = \left\{ 0, \left\lfloor \frac{\ell}{2} \right\rfloor, \ell \right\}.$$

The mixing weights are produced by a small two-layer MLP. Its first layer is shared across mixing sites:

$$\begin{aligned} h_t &= \text{GELU}(W_1 \text{RMSNorm}(x_t^{(\ell)})), \quad W_1 \in \mathbb{R}^{32 \times d}, \\ w_t^{(\ell,r)} &= W_2^{(\ell,r)} h_t + b^{(\ell,r)} \in \mathbb{R}^3, \quad r \in \{\text{attn}, \text{res}\}, \\ \tilde{x}_t^{(\ell,r)} &= \sum_{j \in \mathcal{T}_\ell} w_{t,j}^{(\ell,r)} x_t^{(j)}. \end{aligned}$$

Two mixtures are computed at each site. The first supplies the normalized input to the attention layer:

$$\tilde{x}_t^{(\ell,\text{attn})} = \text{RMSNorm}(\tilde{x}_t^{(\ell,\text{attn})}).$$

The second replaces the residual state onto which the attention output is added:

$$x_t \leftarrow \lambda \tilde{x}_t^{(\ell,\text{res})} + \lambda' y_t.$$

We initialize each second-layer matrix to zero and each bias to select the current residual state:

$$W_2^{(\ell,r)} = 0, \quad b^{(\ell,r)} = (0, 0, 1).$$

Because $x_t^{(\ell)}$ is the final element of $\mathcal{T}_\ell$, the mixer is exactly the identity at initialization. The model therefore begins from the baseline architecture and learns to incorporate earlier residual states progressively throughout training.

This sparse formulation increases step time by only 2.3% while retaining most of the gain observed with the full MUDD formulation. At our scale, it also performed comparably to the higher-overhead configurations of Hyper-Connections [28] and Attention Residuals [29] that we evaluated.

3 PRETRAINING

For the pre-training run of Limite, following the sample efficiency spirit that guided us for the whole work, we adopt very recent optimization algorithms and recipes, taking large inspiration from the Modded Nano GPT speedrun [20]. We pre-train Limite on 6 nodes of 8×H100 [33], for about 449k steps.

3.1 OPTIMIZATION

We optimize the hidden two-dimensional linear-operator weights with NorMuon [34], with the exception of the simplified MUDD and gate linear layers. NorMuon augments Muon’s momentum orthogonalization [35] with neuron-wise adaptive scaling based on an exponential moving average of second-moment statistics. Algorithm 1 gives the complete update used in our final pretraining run.

We use Polar Express [36] for the orthogonalization step and retain Nesterov momentum [37, 38]. We additionally apply Cautious Weight Decay [39] with a base coefficient of 1.2 applied to the squared learning rate η_t^2 , as in the modded-nanogpt optimizer; this is equivalent to scaling the decay with the learning-rate schedule, as proposed in AdamC [40]. This choice was intended to improve training stability and limit excessive weight growth, thereby reducing dynamic-range requirements that can complicate low-precision quantization [41].

All remaining parameters are optimized with AdamW [42]. These parameters are updated once every two training steps, giving this optimizer group twice the effective batch size. AdamW weight decay is disabled for all parameters except the tied token embeddings and the value-embedding table.

We assign separate learning rates, momentum parameters, and weight-decay multipliers to each parameter group. These settings were selected using heuristics and shorter-horizon experimental runs and sweeps, and are reported in Table 1.

Table 1: Per-group optimizer hyperparameters, with Adam WD base $\lambda = 0.005$ and a NorMuon base WD of 1.2, applied as $\eta^2 \cdot \lambda \cdot \text{mult}$

Parameters	Optimiser	Peak LR	Momentum / (β_1, β_2)	wd mult.
W_q, W_k, W_v, W_o	NorMuon	0.018	(0.95, 0.90)	1
$W_{gate}, W_{up}, W_{down}$	NorMuon	0.018	(0.95, 0.90)	1
E (tied)	Adam	0.0019	(0.5, 0.95)	150
$\mathcal{V}$	Adam	0.1400	(0.75, 0.95)	5
scalars	Adam	0.0400	(0.9, 0.99)	0
residual / post lambdas	Adam	0.0016 / 0.0080	(0.9, 0.95)	0
gates and XSA coefficients	Adam	0.0019	(0.9, 0.99)	0
MUDD linears and biases	Adam	0.0019	(0.9, 0.95)	0

Schedules All groups share one learning-rate schedule: a short linear warmup, a nearly flat stable phase, and a linear decay to zero over the last 20% of training. Weight decay scales with η_t^2 and decays with it. NorMuon’s momentum is warmed up from 0.85 to 0.95; batch size and context length are fixed.

Pre-training run The final pre-training run used a batch size of 524k tokens. This relatively small batch was part of our sample-efficiency-oriented training strategy. The run remained stable for approximately 450k training steps, with no major events such as large loss spikes.

Table 2 summarizes the complete model configuration used for pre-training and post-training.

Table 2: Architecture of the Limite base model. Layer indices ℓ start at 0. Parameter counts include the value-embedding table (38.8M); the input and output embeddings are shared and counted once.

Model	
Total parameters	1.04B
Non-embedding parameters	0.80B
Layers	48
Hidden size d	1280
Vocabulary size	151,667
Context length	8192 in pretraining, 131k in post-training
Attention	
Query heads / KV heads	10 / 2
Head dimension	128
Local layers	36, sliding window 1024
Global layers	12 ($\ell \equiv 3 \pmod{4}$), full context
Positional encoding (local / global)	partial RoPE, 64 of 128 dims, base 1024 / NoPE
Feed-forward	
Type	SwiGLU, no bias
Intermediate size d_{ff}	3328 (2.6 d)

3.2 PRE-TRAINING DATA CURATION

The composition of the pretraining corpus plays a central role in defining what it means for the base model to be mathematics-specific. We wanted the pretrained policy to acquire a strong mathematical prior before post-training, while retaining sufficient linguistic variety to understand problems expressed in different styles.

We therefore constructed a high-quality corpus that minimizes non-mathematical content without sacrificing the breadth or volume of mathematical knowledge. The corpus is divided into three source families:

- **Question and Answers (QA):** problem–solution pairs, both synthetic and human-written, with and without explicit reasoning traces.
- **Synthetic Knowledge (SK):** synthetic mathematical documents, primarily written in the style of textbooks, lecture notes, and teacher–student conversations about specific mathematical topics.
- **Filtered Web (FW):** quality-filtered web pages, approximately 40% of which consists of synthetic rewrites of material from the same crawl.

The final mixture contains 46% QA, 19% SK, and 35% FW.

We assessed data quality along several dimensions. These included the estimated fraction of documents containing clear logical or mathematical errors and, for the QA component, the correctness of the reported reference answers. Because exhaustive verification was impractical, we used LLM-based checks on subsamples from each source. In practice for each source we draw uniform random subsamples of documents, and ask an LLM to assign a verdict, together with a written justification; for QA documents the reference answer was independently re-derived and compared. Per-source error rates and confidence intervals were then used to decide inclusion.

QA and SK provide most of the corpus’s explicit mathematical knowledge and reasoning content. FW serves a complementary role by increasing linguistic and stylistic diversity. Its inclusion was carefully filtered to minimize logical errors and common web-data artifacts, including formatting markup and non-English text.

We intentionally included a substantial proportion of explicit reasoning traces. Our aim was to provide a stronger initialization for the subsequent post-training pipeline, motivated by growing controlled evidence that exposing models to diverse reasoning patterns earlier in training can improve both base-model reasoning capabilities and their ability to benefit from subsequent post-training

stages [43, 44, 45, 46, 19].

We applied lexical decontamination followed by a lighter semantic decontamination stage. The semantic filtering threshold was tuned separately for each source family, and both stages were applied against every benchmark reported in this work.

The majority of the corpus has a general knowledge cutoff around the final months of 2025. Most non-synthetic web documents predate 2025, although a small fraction extends into early 2026.

The final polished mixture contains approximately 346 billion tokens. The final pretraining run consumed a 236 billion tokens subset of this mixture, producing the checkpoint that we refer to as *Base*. This checkpoint was then used as the starting point for the post-training pipeline.

3.3 MODEL SOUPING

After pretraining, we experimented with model souping [47], in which checkpoints obtained from related training runs are merged by averaging their parameters. Our objective was to improve upon the base model using a comparatively small amount of high-quality data, without collapsing the model’s output diversity.

We constructed a mixture of approximately 6 billion tokens from the highest-quality sources in the pretraining corpus together with selected post-training data. A large proportion of these documents contain explicit reasoning traces. Starting from the same base checkpoint, we trained three independent runs in parallel, varying the data-shuffling seed and learning-rate settings between runs. We then formed the final soup by uniformly averaging the three resulting checkpoints [48].

As shown in Table 3, the souped model outperforms each individual run and the final pretraining checkpoint on MATH-500 evaluation, and also improves the model’s reasoning capabilities.

We release this final soup together with the *Base* checkpoint used to initialize the post-training pipeline.

Table 3: MATH-500 greedy generation accuracy (%) of the base checkpoint, the three single runs and their uniform soup, with and without thinking. With thinking, the prompt ends with an opening `<think>` token. Coloured numbers give the change from thinking relative to the same model without it. Bold marks the best value within each prompt setting.

Model	Prompt		Δ
	Base	Think	
Limite Base V5	61.8	59.2	−2.6
Single run A	67.4	73.0	+5.6
Single run B	66.2	74.6	+8.4
Single run C	64.0	71.6	+7.6
Uniform soup (A+B+C)	68.4	80.2	+11.8

3.4 BASE-MODEL EVALUATION

Table 4 reports the mathematical-reasoning results available for the base checkpoints before post-training.

With thinking, the soup improves on the base checkpoint on MATH-500 and every AIME year. GSM8K behaves differently: thinking lowers both models to about 46%, and without thinking the soup trails the base by 17 points. We suspect our MATH-style few-shot exemplars, which suit GSM8K poorly, and that the soup is more sensitive to this format mismatch than the base checkpoint.

Table 4: Accuracy (%) of the base checkpoint and of the final soup, without and with thinking. Few-shot prompts use MATH-style exemplars, which may not suit GSM8K: 4-shot for MATH-500 and AIME, 8-shot for GSM8K; thinking appends an opening <think> token. Sampling at $T = 0.6$, top- $p = 0.95$, seed 0, 8,192 generated tokens within a 12,288-token context. Responses per problem: 4 for MATH-500, 1 for GSM8K, 32 for AIME; lenient scoring averaged over all responses.

Model	MATH 500	GSM8K	AIME 2024	AIME 2025	AIME 2026
Limite Base	60.70	80.52	3.33	3.75	3.33
Limite Base (thinking)	73.40	46.10	15.52	23.85	16.88
Limite Base Soup	59.25	63.46	7.40	8.54	8.23
Limite Base Soup (thinking)	81.50	46.32	33.33	32.92	34.06

4 POST-TRAINING

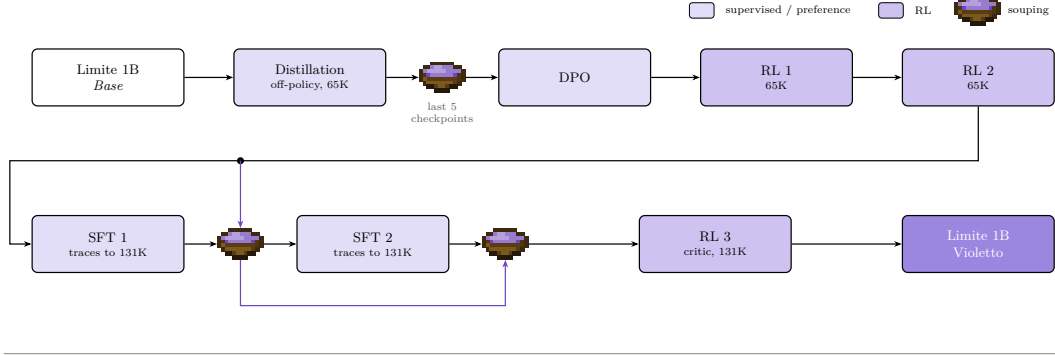


Figure 3: The post-training pipeline, from Limite 1B *Base* to Limite 1B - Violetto. We first distill the base model off-policy at a 65K context, average the last five distillation checkpoints, and apply DPO. Two RL runs follow, also at a 65K context. A first short SFT stage then trains on longer traces of up to 131K tokens, and its final checkpoint is souped with the checkpoint from the second RL run. A second short SFT stage, again on traces of up to 131K tokens, starts from this soup, and its final checkpoint is souped with the first soup. The last RL run starts from the second soup, uses a critic, and trains at a 131K context. Each bowl marks a souping step, a weighted average of checkpoints.

While we find that the base model performs comparatively well with respect to similar base models, it nevertheless has never been exposed to long rollouts nor to its *own* rollouts during training. We deem these as instrumental for downstream performance, and therefore orient our post-training on these directions.

Our post-training pipeline follows the same sample-efficiency goal as pre-training. It begins with a comparatively long knowledge-distillation [49] (KD) stage followed by a short direct preference optimization [50] (DPO) run, then interleaves three reinforcement-learning runs with short, highly curated supervised fine-tuning (SFT) [51, 52] stages and model souping. We view the SFT stages as introducing targeted capabilities, and the RL runs as sharpening them with high learning rates and relatively few expensive updates. Our RL procedure draws from DPPO [53], DAPO [54] and VAPO [55]-related variants.

4.1 POST-TRAINING PIPELINE

Post-training has become a fundamental step of frontier model training, and apart from data, the algorithms used play an essential role in the downstream capabilities of the models. Research on this field recently accelerated and one can now choose from a plethora of solutions when assembling the pipeline. We choose to initially distill from a teacher model and then combine highly curated short SFTs and souping with custom RL algorithms to push the model capability as much as we can, exploiting high learning rates despite being off-policy.

Knowledge distillation The first stage is teacher-forced, logit-level distillation on the corpus described in Section 4.2. At each position, the training target is the teacher’s top 32 log-probabilities for its generated solution. We conclude by uniformly averaging the final five checkpoints [56].

Direct preference optimization After distillation, we apply one epoch of offline DPO [50] with mean (length) normalization of the implicit reward, following LN-DPO [57]. Both the trained policy and the frozen reference policy are initialized from the distilled soup. Averaging the per-token log-ratios, rather than summing them, keeps the preference signal from favouring shorter or longer responses (Appendix E). We train for 140 updates of 128 preference pairs each, with $\beta = 5$, using AdamW [42] with a peak learning rate of 3×10^{-6} and linear warmup and decay. No sequence is truncated: every pair fits within the 65,536-token context.

Supervised fine-tuning and checkpoint averaging For SFT, we adapt the pretraining process to longer contexts while retaining optimizer and hyperparameters configurations, with the exception of the peak learning rate which we set to 15% of the peak used in pre-training. Targeted mixtures and checkpoint averaging make these short stages effective. As outlined below, we interleave multiple RL and SFT phases. During the two post-RL SFT stages, we form one model soup between the stages and another before the final critic-assisted RL run. Each is an average of the current and preceding checkpoints [47]. These stages also extend the maximum context length from 65k to 131k tokens.

Interleaving SFT and RL Continued RL eventually became unstable in our runs since we adopted a high learning rate, high off-policy regime. Since simply lowering the learning rate would worsen the sample efficiency of the already expensive updates, we opted for short SFT stages to stabilize trajectories while preserving capabilities acquired through RL. SFT introduces targeted information and raises $\text{pass}@k$ [58, 59], but can also move the model away from its on-policy distribution and weaken coherent sampling from that distribution. We found that averaging the pre- and post-SFT checkpoints balances these new capabilities against the sampling behaviour learned through RL; Following runs then continued from the resulting soup.

4.2 POST-TRAINING DATA CURATION

To post-train a compact 1B-parameter model with substantially fewer tokens than models with similar math capabilities, we prioritize data quality across three sources: distillation targets, problems for RL, and problem–solution pairs for the short SFT stages.

Distillation corpus The knowledge-distillation corpus contains 5.5 million solutions generated by our main teacher, VibeThinker-3B [60], for publicly sourced mathematical problems. These solutions comprise approximately 26 billion target tokens. At every position, the student is trained against the top 32 log-probabilities from the teacher distribution.

Public datasets may contain incorrect gold answers, damaged examples, or invalid solutions. We therefore apply three filters:

- We retain a trace only if it reaches the problem’s gold answer.
- We discard examples that are not mathematical problems or that arrive malformed.
- We discard traces in which the teacher abandons the derivation and guesses, identified through phrases such as “given the time constraints” or “I’ll just go with.”

The first filter removes incorrect traces and most examples with incorrect gold answers, but not cases in which the teacher happens to match an incorrect gold answer. We pass rejected traces to a judge model, which receives the problem and trace without the gold answer and determines whether the problem is well posed and the solution correct. This recovers 18% of the corpus tokens that the first filter would otherwise reject.

Solutions contain fewer than 5k tokens on average and are capped at 65,536 tokens. Approximately one third of all tokens lie in a long-tail subset whose median solution length is around 15k tokens; the distillation schedule ends on this long-tail portion.

Targeted data selection Distillation provides an initially broad exposure to general mathematical problem solving; the subsequent bottleneck then remains performance on specific topics. We identify these topics through failure analysis and area-level accuracy, and direct later data generation toward the identified weaknesses. The short SFT stages consequently use an order of magnitude fewer tokens than distillation, concentrating on capabilities the current checkpoint does not yet exhibit reliably.

Preference pairs For the preference-optimization stage we use the mathematics subset of Dolci-Think-DPO-32B [61], whose pairs follow the delta-learning construction [62]: the chosen response is generated by Qwen3-32B and the rejected response by Qwen3-0.6B, both in reasoning mode. We decontaminate following the same procedure followed in pre-training, described in Section 3.2. We apply this same decontamination procedure to all of our post training work.

RL problem generation Rather than relying exclusively on public datasets, we build a dedicated generation pipeline that controls problem difficulty and schedules themes and mathematical areas across runs. Because an incorrect gold answer rewards mathematical errors and penalizes correct reasoning, every generated problem must have a programmatically computed answer and an empirical difficulty estimate from the checkpoint being trained. We use two complementary generators: an LLM-based generator and a CPU-based procedural generator.

The LLM-based generator produces a problem statement, a solving program, and a claimed answer. A validation gate re-executes the program in a separate environment and retains the problem only if its output matches the claimed answer. The generation request selects a mathematical area from a fixed list of subcategories, supplies a small set of examples, and asks for an unambiguous statement and a program implementing its conditions. Since the examples have the strongest influence on difficulty and register, the initial waves use a small, carefully curated set; later waves add generated examples whose measured difficulty falls in the useful range. Areas are sampled non-uniformly, concentrating generation on weaker areas. We outline the LLM-based generation process in Figure 4.

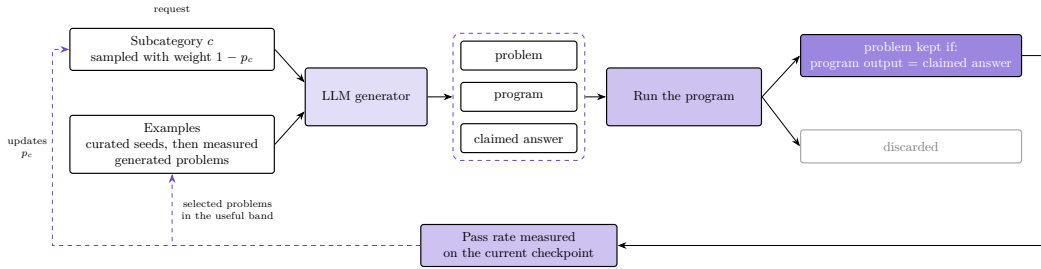


Figure 4: Graphical representation of the LLM-based generator.

The complementary procedural generator is fully CPU-based and inspired by [63]. This is suitable for RL, where thematic variety matters less than for SFT because GRPO [64] baseline subtraction removes signals shared across a rollout group. The generator combines high-quality program templates with keyword synonyms and alternative phrasings, preserving programmatic guarantees while adding linguistic variation. Each wave is strictly filtered and deduplicated; sampling hyperparameters are tuned per template, and a tuned sampler discourages trivial strategies such as direct enumeration in combinatorics problems. We outline the procedural generation process in Figure 5.

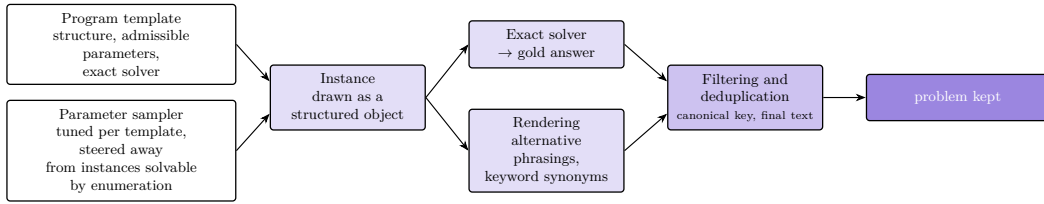


Figure 5: Diagram explaining the procedural generator.

The two generators offer complementary trade-offs. Once implemented, a procedural family can sample, solve, and render an instance in milliseconds on CPU, making volume inexpensive and limiting each wave primarily by the number of available families. By contrast, LLM-based generation may require tens of thousands of GPU-generated tokens per problem, but it can vary the underlying mechanism rather than only details and form. We therefore use procedural generation for volume and controllable difficulty, and LLM-based generation for structural variety.

After applying the same error analysis done on the public datasets, the generated sets yield a substantially lower rate of incorrect gold answers, approaching zero in our measurements.

Difficulty calibration Group-relative RL is informative only when the policy solves a problem intermittently: groups in which every rollout succeeds or every rollout fails have zero relative advantage. We therefore estimate pass rates using groups of eight rollouts from a fixed checkpoint and

retain an intermediate difficulty band, favouring problems near a 50% pass rate. For each wave, we analyze a representative sample large enough to estimate the difficulty distribution across families and subcategories; the result shifts subsequent sampling toward areas in which the current policy remains weaker [65, 66, 67].

Short-SFT mixtures By the end of the second RL run, reward had stopped improving, and failure analysis showed that the model remained weak in specific mathematical areas. Because we believe RL in our setting tends to only reinforce behaviors already represented by the policy [68, 69], we introduced two short SFT stages with different mixtures of the following two categories of sources:

- **Synthetic problems targeting the areas in which the current policy is weakest.** The solutions are generated by Qwen3.8-27B and Qwen3.8-Flash-Next [70, 71] with a context length of 131k tokens.
- **The remainder combines solutions generated by the current policy for our synthetic problems,** which limit drift from the RL checkpoint, with a small proof subset. The proof subset is the union of proofs from public datasets and synthetic proofs for publicly available mathematical data, with the latter verified by an LLM judge. We include these proofs to introduce reasoning capabilities not yet present in the current policy.

4.3 REINFORCEMENT-LEARNING METHOD

Reinforcement learning is central to reasoning-model post-training but expensive because rollouts are generated autoregressively. For Limite, the 48-layer architecture also gives a relatively high per-token latency despite its small parameter count. We therefore optimize for wall-clock efficiency through the following training choices:

- **High learning rates** We use learning rates from 1×10^{-5} to 2×10^{-5} , compared with the 1×10^{-6} scale commonly used in comparable RL runs.
- **Many off-policy updates** We perform between 7 and 19 updates per behaviour policy, using the DPPO mask to control policy drift.
- **Few actor updates** We perform 461 RL updates across the three runs, rather than the thousands of updates often used in larger-scale training.
- **Interleaved SFT and RL** Short SFT stages introduce targeted capabilities and allow RL to resume from a different checkpoint.
- **Critic-assisted credit assignment** After the policy has learned the easiest improvements available through group-relative training, we introduce a critic for the final RL run.

The first two runs use the group-mean advantage estimate associated with GRPO [64]; the third uses a learned value model to assign token-dependent advantages. The credit-assignment and critic paragraphs below motivate this transition, followed by the complete training sequence.

Objective, reward, and dynamic sampling Let o_i be a response generated by policy π_θ for query q_i . We use a sparse terminal reward at token t :

$$R_{i,t} = \begin{cases} 1, & \text{if } t = |o_i| \text{ and } o_i \text{ has a correct final answer,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

For each problem, we sample G rollouts and apply dynamic sampling, introduced by DAPO [54], so that groups with pass rate $0/G$ or G/G are discarded because all their group-relative advantages are zero. The resulting batches contain only groups that provide a learning signal.

To maximize the number of off-policy updates while retaining training stability, we combine the DPPO masking mechanism with CISPO [72]-style clipping of the importance sampling weight. For rollout j associated with query i , define

$$r_{i,j,t} = \frac{\pi_\theta(o_{i,j,t} \mid q_i, o_{i,j,<t})}{\mu(o_{i,j,t} \mid q_i, o_{i,j,<t})},$$

where μ is the behaviour policy. Let $\hat{A}_{i,j,t}$ be the advantage and $M_{i,j,t}$ the DPPO mask. The token-normalized policy loss is

$$\mathcal{L}_{\text{policy}}(\theta) = \frac{\sum_{i=1}^B \sum_{j=1}^G \sum_{t=1}^{|o_{i,j}|} M_{i,j,t} \hat{A}_{i,j,t} \tilde{r}_{i,j,t}}{\sum_{i=1}^B \sum_{j=1}^G |o_{i,j}|}. \quad (2)$$

Here, $M_{i,j,t} = 1$ includes the token and $M_{i,j,t} = 0$ masks it. The quantity $\tilde{r}_{i,j,t}$ has the same forward value as $r_{i,j,t}$ but a clipped gradient. We normalize by tokens rather than samples [54]; both choices introduce length-dependent biases, and for these runs we prefer the token-level bias.

Rollout batching We considered two ways to overlap rollout generation and training:

1. As soon as every rollout in a group finishes, place the group in a buffer if it is informative and its behaviour policy is sufficiently recent. When the buffer is full, update the target policy while generation continues.
2. Generate a cohort of B complete groups, discard non-informative groups, split the remainder into minibatches, and then update on those minibatches.

The first approach is faster, but it preferentially admits groups that finish early. This creates two potential biases:

- **Length bias** Training more often on short rollouts may encourage the model to shorten its reasoning.
- **Problem-difficulty bias** Problems solved quickly are often easier, so early admission may overrepresent easier problems.

Both approaches overlap rollout generation and training, the only difference is how rollouts enter in a batch. The first one is similar to PRIME-RL’s [73] asynchronous buffering, while the second one is closer to DAPO’s [54] implementation. We implement the second approach using double buffering, as illustrated in Figure 6, to obtain more representative cohort-level updates. Our double buffering setup resembles those of Asynchronous RLHF [74] and Cleanba [75].

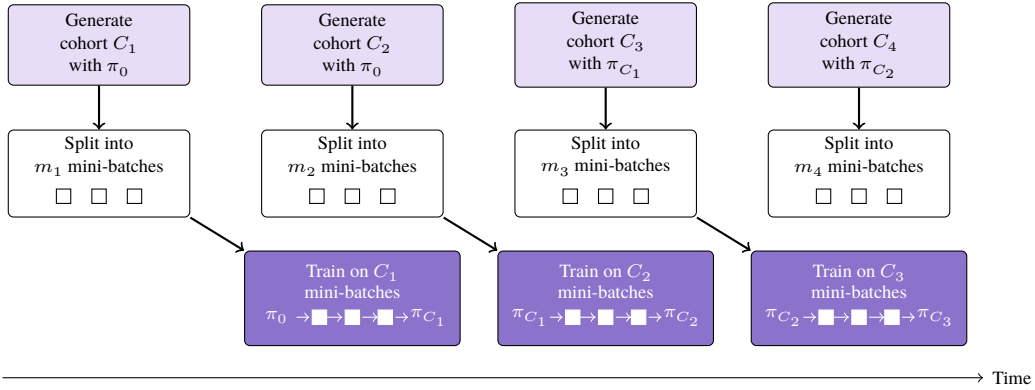


Figure 6: Double buffering across cohorts. C_k denotes the k -th cohort and m_k the number of mini-batches obtained from it. The white squares represent mini-batches schematically. The trainer performs one update per mini-batch while the next cohort is generated. π_0 is the initial policy, and $\pi_{C_{k+1}}$ is the policy π_{C_k} after completing m_{k+1} updates on C_{k+1} mini-batches.

As shown in Figure 6, unless from the initial policy π_0 , each policy trains on off-policy rollouts. The lag between the target policy $\pi_{C_{k+2}}$ and the behaviour policy π_{C_k} coincide with $m_{k+2} + m_{k+1}$ for each integer $k > 1$.

Off-policy optimization and DPPO masking In our setting inference is the main bottleneck. Since all rollouts in a cohort must finish before we can update the policy on that cohort, the longest rollouts largely determine generation time. When inference capacity is available, generating more rollouts in parallel can increase throughput without significantly increasing generation time.

We therefore aim to obtain more policy updates from each generated cohort, amortizing the cost of rollout generation across those updates. At a fixed minibatch size, a larger cohort provides more minibatches and therefore more updates. Alternatively, reducing the minibatch size allows us to perform more updates on the same cohort without generating additional rollouts.

However, performing many off-policy updates leads the target policy to drift from the behaviour policy. As this drift grows, the variance of importance sampling weights increases, making only few tokens dominate the gradient contribution. Hence, the policy gradient estimates become noisier. To support multiple off-policy updates while limiting this drift, the importance ratio is conventionally clipped to a fixed interval in PPO [76] and GRPO [64],

$$\text{clip}\left(\frac{\pi_\theta(o_t | q, o_{<t})}{\mu(o_t | q, o_{<t})}, 1 - \epsilon, 1 + \epsilon\right).$$

In practice, this clipping is asymmetric in terms of tokens: rare tokens can leave the interval after relatively few updates, whereas already-likely tokens require a larger change. The resulting gradient suppression may penalize exploration more strongly than exploitation and contribute to entropy collapse. Previous works like DPPO [53] and DAPO [54] already discussed these issues.

Instead, we therefore use the DPPO mask. Given the context at token t , we can approximate the pointwise divergence between target- and behaviour-policy probabilities as

$$D_{i,j,t} = |\pi_\theta(o_{i,j,t} | q_i, o_{i,j,<t}) - \mu(o_{i,j,t} | q_i, o_{i,j,<t})|.$$

Let δ be the trust-region radius. The mask is

$$M_{i,j,t}^{\text{DPPO}} = \begin{cases} 0, & \text{if } (\hat{A}_{i,j,t} > 0 \wedge r_{i,j,t} > 1 \wedge D_{i,j,t} > \delta) \\ & \text{or } (\hat{A}_{i,j,t} < 0 \wedge r_{i,j,t} < 1 \wedge D_{i,j,t} > \delta), \\ 1, & \text{otherwise.} \end{cases}$$

The mask removes only tokens for which the probability difference exceeds δ and the gradient update would move the target policy still farther from the behaviour policy.

Credit assignment Credit assignment governs the variance and specificity of the policy-gradient estimate. Appendix B shows that, with our settings, REINFORCE [77] without a baseline has the same update direction as online rejection fine-tuning. The main benefit of GRPO- or PPO-style optimization over online RFT [78] in this setting is therefore the variance reduction supplied by a baseline. A common choice is the state value $V^\pi(s_t)$; Appendix C defines value functions and advantages, and Appendix D summarizes the Generalized Advantage Estimation (GAE) [79] used by actor-critic PPO with a learned value model V_ϕ^π .

Training a value model adds cost. We therefore use the GRPO group-mean baseline, without reward normalization and following the unnormalized treatment used in Dr. GRPO [80], for the first two RL runs. Let q be a query, G the group size, and $\{o_i\}_{i=1}^G \sim \pi(\cdot | q)$ the sampled responses. The advantage assigned to every token in response o_i is

$$\hat{A}_{i,t} = R_{|o_i|} - \bar{R}, \quad \bar{R} = \frac{1}{G} \sum_{j=1}^G R_{|o_j|}. \quad (3)$$

The group-average reward $\bar{R}$ is a Monte Carlo estimate of the problem-level value, and every token in a rollout receives the same advantage as if all visited states had that value.

This estimate has two limitations:

1. **Problem value as the baseline** If state values change substantially over a rollout, the problem-level value may be a poor baseline for many individual states. See the Figure 7 to see an illustration describing a similar situation.

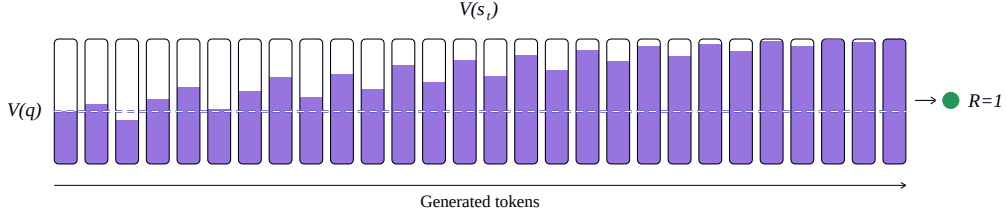


Figure 7: Schematic illustration of state values along a successful rollout. The dashed line denotes the problem-level baseline $V(q)$.

2. **The same advantage for every action** If only a small number of decisions determine the final outcome, assigning the same advantage to every action provides a diffuse training signal.

The first limitation should matter more for long rollouts, where more visited states may differ substantially from the problem-level value. We hypothesize that the second becomes more important as the policy improves: a failure in an otherwise reliable derivation may stem from only a few decisions, yet the group-level estimator cannot distinguish them from the remaining tokens. These considerations motivate the value model used in the third RL run.

Critic model Inspired by Pan, Zhou, Li, et al. [81], we initialize the value model from the actor and replace the unembedding head with a scalar value head. Let s_t be the rollout prefix through token t , h_t the critic’s final hidden state for that prefix, and w and b the value-head parameters. The state-value estimate is

$$V_\phi(s_t) = w^\top h_t + b. \quad (4)$$

Thus, the head is a scalar linear projection without an activation function, as in [81].

We warm up the critic, in two phases, for a total of 501 updates:

1. **First phase** We train the critic for 225 updates on already available rollouts previously generated from older actor policies. In the meantime, we start generating rollouts with the best actor checkpoint until that point, which we use as the starting checkpoint of the third RL run.
2. **Second phase** Once those rollouts are available, we train the critic for another 276 updates on responses generated by that checkpoint.

Throughout both phases, the actor remains frozen and only the critic is trained.

We also apply dynamic sampling [54] to the critic data. In fact, if we included too easy or too hard problems, we would encourage the critic to learn only problem difficulty rather than tracking state values under the actor policy. For stability, we estimate on a large sample the constant output bias that minimizes the initial loss.

For critic training¹, we set $\gamma_{\text{critic}} = \lambda_{\text{critic}} = 1$. The target for state s_t is therefore the terminal reward of the rollout that continues from that state. We train with the token-averaged mean-squared error

$$\mathcal{L}_{\text{critic}}(\phi) = \frac{\sum_{i=1}^B \sum_{j=1}^G \sum_{t=1}^{|o_{i,j}|} (V_\phi(s_{i,j,t}) - R_{i,j})^2}{2 \sum_{i=1}^B \sum_{j=1}^G |o_{i,j}|}. \quad (5)$$

As shown in the warm-up curves in Figure 8, the second phase converges within a few updates. Nevertheless, the critic reduces downstream policy-gradient variance, which helps stabilize training, so we extend the second phase beyond what would otherwise appear to be a more natural stopping point. We refer readers to Section 4.4 for the description of the third RL run, in which we use the critic.

¹See Appendices C and D for the definitions of value functions, advantages and GAE.

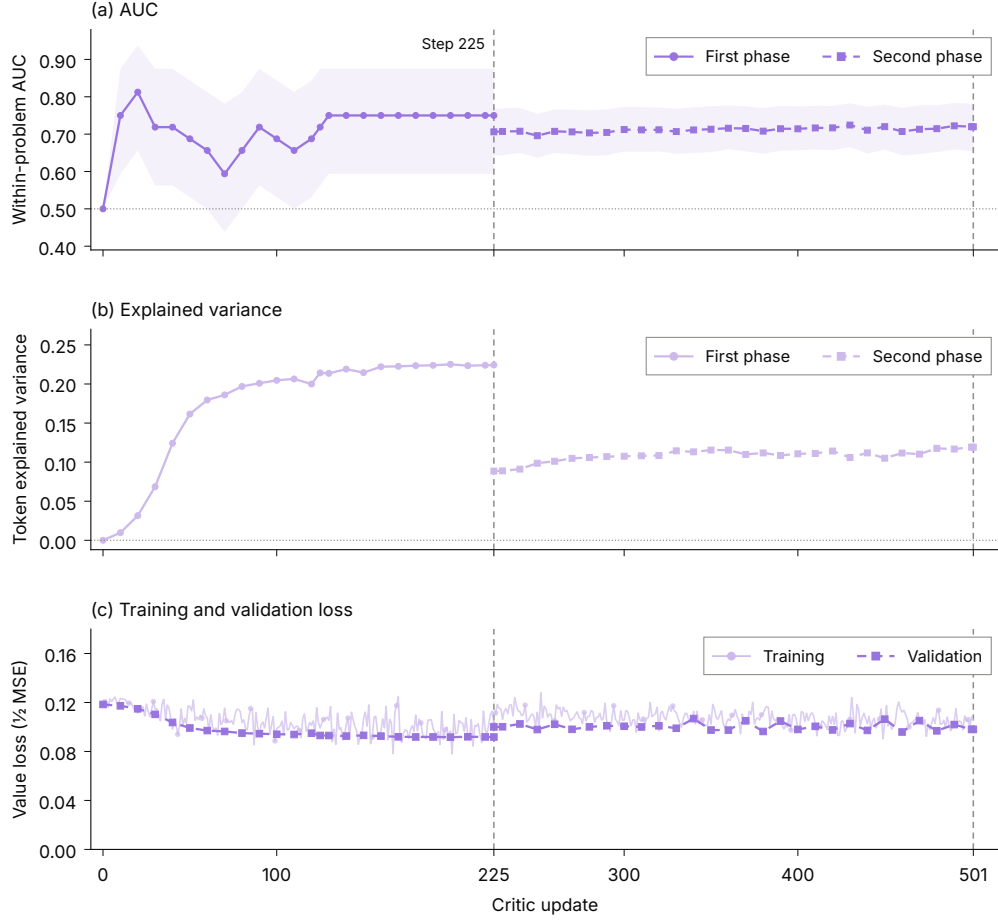


Figure 8: Value-model metrics during offline warm-up: (a) AUC on validation subset, (b) explained variance on validation subset, and (c) training and validation loss. All the plots share the x-axis that tracks the number of critic updates. These metrics are useful to evaluate the critic performance. AUC tells us about how able the critic is at ranking correct rollouts above incorrect ones, whereas explained variance measures the fraction of return variance removed by using the critic as a baseline. The dashed lines mark the phase transition at update 225 and the end of warm-up at update 501.

4.4 RL TRAINING SEQUENCE AND RESULTS

We perform three DPPO runs, all with AdamW for both actor and critic (present only in the third run). Throughout, we set the DPPO trust-region radius to $\delta = 0.15$, and clip the importance-sampling weight at 5.

First RL run The first run tests whether a high learning rate and a large number of off-policy updates can produce rapid gains. By update 213, we found a good balance of performance and brevity as shown in Table 5 and Table 6. Beyond that, excessive drift between target and behaviour policies produces a sharper optimization regime and eventual reward collapse, so we select checkpoint 213. Each cohort contains 704 problems with 8 rollouts each. After discarding non-informative groups, we split the remainder into minibatches of 32 problems; because the retained count varies by cohort, so does the number of optimizer updates. Responses are capped at 61,440 tokens despite the preceding model’s exposure to longer traces, both to encourage concise reasoning and to reduce rollout time. We perform 19 off-policy updates per behaviour policy on average at a constant learning rate of 1.5×10^{-5} .

Table 5: Benchmark accuracy (%) across post-training stages. Off-policy distillation, DPO pre-RL, First RL, and Second RL were evaluated with a context limit of 65,536 and a response-token cap of 61,440. Columns marked † (SFT post-RL and Third RL) were evaluated with a context limit of 131,072 and a response-token cap of 126,976, following the decision to extend the model’s context length and train for longer reasoning. Gains are percentage-point differences from the preceding stage, computed from displayed scores. Detailed results at 65K and 131K context lengths are reported in Appendix G.

Benchmark	Off-policy distillation	DPO pre-RL		First RL		Second RL		SFT post-RL [†]		Third RL [†]	
	Score	Score	Gain	Score	Gain	Score	Gain	Score	Gain	Score	Gain
AIME 2025	77.08	80.52	+3.44	85.62	+5.10	87.71	+2.09	89.17	+1.46	90.21	+1.04
AIME 2026	81.67	83.54	+1.87	87.50	+3.96	87.40	−0.10	91.35	+3.95	94.01	+2.66
HMMT Feb. 2025	70.42	76.15	+5.73	80.73	+4.58	80.62	−0.11	89.69	+9.07	91.35	+1.66
HMMT Feb. 2026	64.68	62.41	−2.27	73.01	+10.60	72.25	−0.76	80.78	+8.53	83.62	+2.84
APEX Shortlist	23.27	21.81	−1.46	33.44	+11.63	32.05	−1.39	43.42	+11.37	50.80	+7.38
BeyondAIME	57.16	52.25	−4.91	65.28	+13.03	65.34	+0.06	72.38	+7.04	74.25	+1.87
ArXivMath May 2026	20.08	15.39	−4.69	25.08	+9.69	24.14	−0.94	23.52	−0.62	25.08	+1.56

Table 6: Mean generated response length (tokens) at a context limit of 131,072 and a response-token cap of 126,976.

Benchmark	Off-policy distillation	DPO pre-RL	First RL	Second RL	SFT post-RL	Third RL
AIME 2025	27,725	35,506	22,736	21,226	23,474	31,968
AIME 2026	24,219	32,800	19,502	18,376	21,121	29,863
HMMT Feb. 2025	35,037	42,709	26,973	24,393	28,309	39,647
HMMT Feb. 2026	37,825	44,842	28,404	26,815	33,042	44,297
APEX Shortlist	50,485	59,860	38,747	32,359	54,239	75,007
BeyondAIME	39,323	49,517	30,540	27,404	36,936	52,387
ArXivMath May 2026	40,908	65,128	34,097	29,474	46,560	68,002

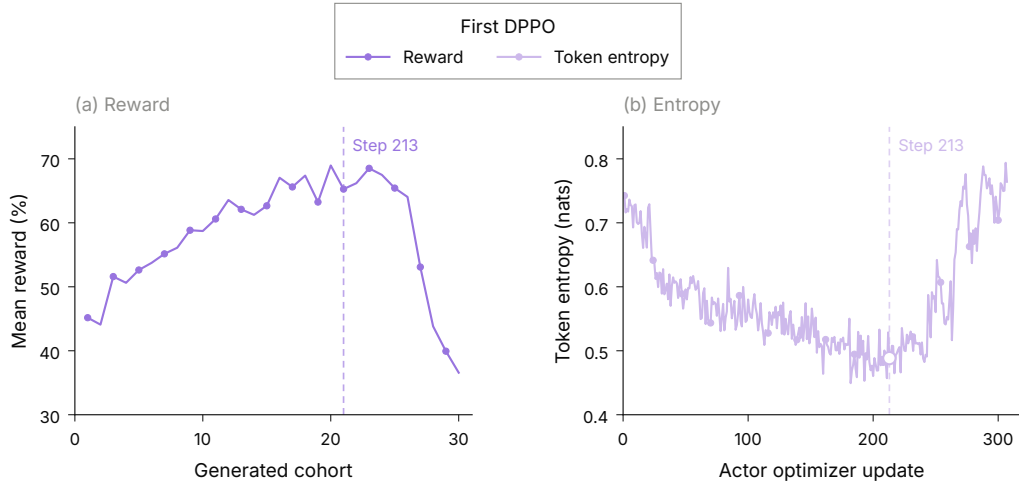


Figure 9: First DPPO: complete training trajectory, including continuation beyond the selected checkpoint. Left: mean binary correctness reward across all responses in complete generated cohorts, including truncated responses. Right: mean token entropy on informative training minibatches. Dashed lines mark checkpoint 213.

Second RL run The second run starts from the first run’s selected checkpoint. To improve stability, we double the minibatch size, thereby reducing the number of off-policy updates, and add data to broaden the training signal. More details about the data we added are provided in the paragraph *RL problem generation* in Section 4.2. Performance improves slightly by update 179, after which target-behaviour divergence again leads to reward collapse; we therefore select checkpoint 179. Cohort and group sizes remain 704 problems and 8 rollouts, while the minibatch increases to 64 problems thereby reducing the mean number of off-policy updates per behaviour policy from 19 to 9. We begin at 1×10^{-5} and, because reward improves too slowly, increase the learning rate to 2×10^{-5} after 73 actor updates. We also reduce the gradient-norm clipping threshold to 0.02 to limit spikes that could drive the target policy too far from the behaviour policy. The response cap remains 61,440 tokens.

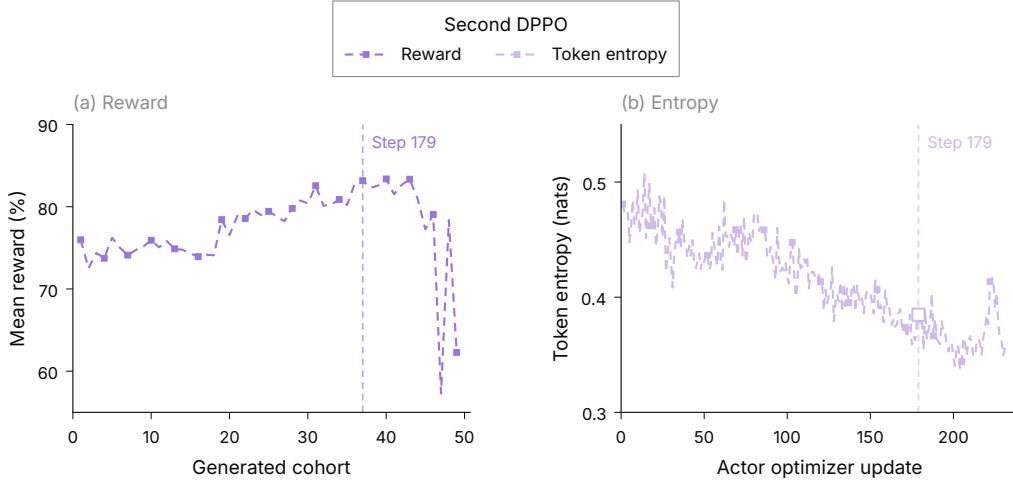


Figure 10: Second RL run: complete training trajectory, including continuation beyond the selected checkpoint. Left: mean binary correctness reward across all responses in complete generated cohorts, including truncated responses. Right: mean token entropy on informative training minibatches. Dashed lines mark checkpoint 179.

Intermediate SFT and model soup The second run’s limited improvement and eventual instability suggested that further RL on the same trajectory would be inefficient. As we discuss in the paragraph *Interleaving SFT and RL* in Section 4.1, training through SFT is more efficient in these cases. We therefore perform two short SFT stages on the targeted mixture described above, averaging checkpoints once between them and again after the second stage. The resulting soup initializes the final RL run. This stage also expands the supported context length from 65k to 131k tokens; its benchmark effect appears in the SFT post-RL column of Table 5.

Third RL run The third run starts from the post-SFT soup and uses the learned critic for token-dependent advantages. The model improves substantially within 59 actor updates. We attribute this to the critic together with the stronger initialization produced by targeted SFT and checkpoint averaging, rather than to the critic alone. For the actor, we set $\gamma_{\text{policy}} = 1$ and

$$\lambda_{\text{policy}} = 0.5^{1/L},$$

where L is the rollout length. Under the indexing used in our implementation, this choice is intended to keep the contribution of the terminal reward to the earliest token constant with respect to L :

$$\lambda_{\text{policy}}^L = 0.5.$$

For the critic, we use $\gamma_{\text{critic}} = \lambda_{\text{critic}} = 1$, as described above.

We increase the response cap to 126,976 tokens to train long-context reasoning. The cohort contains 256 problems, the minibatch contains 32 problems, and we perform 7 off-policy updates per

behaviour policy on average. The actor uses a constant learning rate of 1×10^{-5} . For every actor update, the critic receives two updates at a learning rate of 5×10^{-6} so that it can track the changing policy.

The policy objective is otherwise the same as in the first two runs, but the advantages are estimated by the critic rather than by the group-mean baseline. Appendix D gives the corresponding GAE definition.

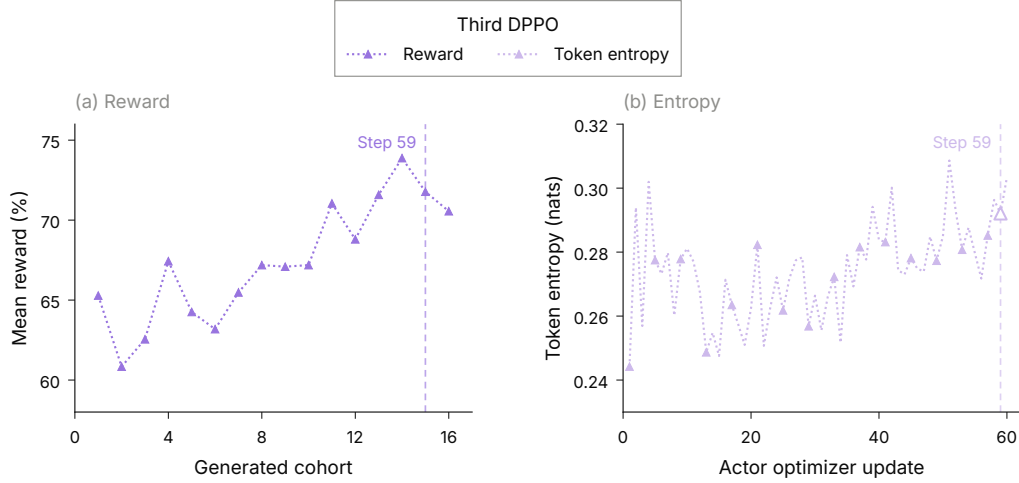


Figure 11: Third RL run: complete training trajectory, including continuation beyond the selected checkpoint. Left: mean binary correctness reward across all responses in complete generated cohorts, including truncated responses. Right: mean token entropy on informative training minibatches. Dashed lines mark checkpoint 59.

5 EVALUATION

We evaluated Limite on seven mathematical reasoning benchmarks: AIME 2025 and 2026, HMMT February 2025 and 2026, the MathArena Apex Shortlist, BeyondAIME, and ArXivMath (May 2026). This suite combines recurring competitions, harder olympiad collections, and research-level problems from recent arXiv papers, while keeping a single property across all of them: every answer can be extracted and verified automatically.

This requirement allows every completion to be graded by deterministic, rule-based comparison against the reference answer rather than by an LLM judge. More importantly, it allows each completion to be scored under increasingly tolerant answer-extraction rules, ranging from a *strict* rule that accepts only a well-formed final `\boxed{}` answer, through a *lenient* rule that recovers plausible final answers written in the wrong form, to a *permissive* rule that accepts the correct value anywhere in the completion. We also track the gaps between these three scores, together with the rates of truncated and malformed outputs.

Across pre-training, supervised fine-tuning and reinforcement learning, these signals show how much of a change in performance comes from genuine mathematical ability and how much comes from a change in output format.

For the headline scores reported in Table 7, we rely on official graders wherever they exist. For AIME 2025 and 2026, HMMT February 2025 and 2026, and the Apex Shortlist, completions are scored with MathArena’s published grader. BeyondAIME does not release an executable grader, and for ArXivMath we found that the MathArena grader produced both false-positive and false-negative verdicts; for these two benchmarks we therefore extract the final `\boxed{}` answer and compare it with the reference using Math-Verify [82]. We release our evaluation suite implementation at github.com/paradigma-inc/limite-violetto.

Table 7: Results on mathematical reasoning benchmarks (pass@1, %, averaged over 32 samples per problem). Models above the line have at most 9B total parameters, and **bold** marks the best score in this group, together with any score whose standard-error interval overlaps it (Table 10). Models below the line are larger models included for reference. Unless marked, all scores come from our own evaluation: AIME, HMMT and the Apex Shortlist are graded with MathArena’s published grader, while BeyondAIME and ArXivMath are graded with Math-Verify on the final `\boxed{}` answer. † Score taken from the MathArena leaderboard or from the official model card instead of being reproduced by us.

Model	Release	Params. total	AIME 2026	HMMT Feb. 2026	APEX Shortlist	Beyond AIME	AIME 2025	HMMT Feb. 2025	ArXivMath May 2026
Limite 1B - Violetto	2026-09	1B	94.01	83.62	50.80	74.25	90.21	91.35	25.08
VibeThinker-1.5B	2025-11	1.5B	70.94	47.25	10.84	48.06	72.92	50.52	18.13
MiniCPM5-2B	2026-09	2B	90.21	67.80	26.86	60.59	86.98	78.02	21.80
VibeThinker-3B	2026-06	3B	93.85	78.98	46.41	72.00	92.19	88.23	29.77
Qwen3.5-4B	2026-02	4B	89.66 †	72.86 †	32.51 †	61.25	82.19	74.58	26.33
Olmo-3-7B-Think	2025-11	7B	69.38	41.95	13.16	41.00	68.02	42.60	18.05
Olmo-3-7B-Instruct	2025-11	7B	44.27	31.91	6.18	22.81	39.38	27.92	11.33
OLMo-Hybrid-Think-SFT-7B	2026-02	7B	62.08	35.89	10.57	36.41	53.65	35.10	15.47
Qwen3.5-9B	2026-02	9B	90.42	70.36	30.72	65.56	88.75	83.02	31.64
Qwen3.5-27B†	2026-02	27B	91.67	81.06	52.13	—	—	—	—
Qwen3.6-27B†	2026-04	27B	94.10	84.30	—	—	—	93.80	—
MUSE-Glimmer-30B	2026-08	30B	93.44	80.87	51.66	70.00	92.71	90.21	28.91
Gemma-4-31B-IT	2026-03	31B	89.17	77.08	40.16	71.00	87.81	86.88	35.78
Qwen3.6-35B-A3B†	2026-04	35B	92.70	83.60	—	—	—	90.70	—
NVIDIA-Nemotron-3-Super†	2026-03	120B	91.67	84.85	57.45	—	—	—	—
GLM 5.2†	2026-06	753B	99.20	92.50	68.09	—	—	—	56.67
Gemini 3 Pro (preview)†	2025-11	n.d.	91.67	86.36	66.49	—	95.00	97.50	—

6 CONCLUSION

We presented Limite 1B - Violetto, a model trained from scratch with the objective of maximizing mathematical capability per parameter. We oriented our decisions across the entire stack, from architecture, to data curation, to optimization and post-training around the goal of maximizing sample efficiency. A mathematics-focused pretraining corpus provided the foundation for subsequent distillation, targeted supervised fine-tuning, and reinforcement learning with verifiable rewards. The resulting one-billion-parameter model is competitive with substantially larger models on hard mathematical problems. Its compact size also supports efficient inference, although its deep architecture can limit decoding speed and the model has not been fully optimized for inference.

This specialization has clear limitations. Limite is only lightly instruction-tuned, and its mathematical performance does not imply comparable capability in general instruction following or other domains. In broader research workflows, it could serve as a mathematical reasoning component within multi-agent systems, with complementary models and tools providing coordination, capabilities outside mathematics, and verification.

Limite 1B - Violetto is our first step towards building models for autonomous science, and will act as a concrete base for our upcoming work. We are already developing the next models and systems around the objective that motivates this effort: maximizing discovery per unit of computation.



REFERENCES

- [1] Mathematical Association of America. *American Invitational Mathematics Examination (AIME)*. <https://maa.org/maa-invitational-competitions/>. (Visited on 10/06/2026).
- [2] ByteDance Seed. *BeyondAIME: Advancing Math Reasoning Evaluation Beyond High School Olympiads*. 2025. URL: <https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME> (visited on 10/06/2026).
- [3] Jasper Dekoninck et al. *Beyond Benchmarks: MathArena as an Evaluation Platform for Mathematics with LLMs*. 2026. arXiv: 2605.00674 [cs.CL]. URL: <https://arxiv.org/abs/2605.00674>.
- [4] Amirhossein Kazemnejad et al. *The Impact of Positional Encoding on Length Generalization in Transformers*. 2023. arXiv: 2305.19466 [cs.CL]. URL: <https://arxiv.org/abs/2305.19466>.
- [5] Bowen Yang et al. *Rope to Nope and Back Again: A New Hybrid Attention Strategy*. 2025. arXiv: 2501.18795 [cs.CL]. URL: <https://arxiv.org/abs/2501.18795>.
- [6] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. arXiv: 1706.03762. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
- [7] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: <https://arxiv.org/abs/2302.13971>.
- [8] Biao Zhang and Rico Sennrich. *Root Mean Square Layer Normalization*. 2019. arXiv: 1910.07467 [cs.LG]. URL: <https://arxiv.org/abs/1910.07467>.
- [9] Ofir Press and Lior Wolf. *Using the Output Embedding to Improve Language Models*. 2017. arXiv: 1608.05859 [cs.CL]. URL: <https://arxiv.org/abs/1608.05859>.
- [10] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL]. URL: <https://arxiv.org/abs/2004.05150>.
- [11] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: <https://arxiv.org/abs/2310.06825>.
- [12] Rewon Child et al. *Generating Long Sequences with Sparse Transformers*. 2019. arXiv: 1904.10509 [cs.LG]. URL: <https://arxiv.org/abs/1904.10509>.
- [13] Gemma Team. "Gemma 2: Improving Open Language Models at a Practical Size". In: *arXiv preprint arXiv:2408.00118* (2024).
- [14] Gemma Team et al. *Gemma 3 Technical Report*. 2025. arXiv: 2503.19786 [cs.CL]. URL: <https://arxiv.org/abs/2503.19786>.
- [15] Mohammad Aflah Khan et al. *Fractional Rotation, Full Potential? Investigating Performance and Convergence of Partial RoPE*. 2026. arXiv: 2603.11611 [cs.LG]. URL: <https://arxiv.org/abs/2603.11611>.
- [16] Jianlin Su et al. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL]. URL: <https://arxiv.org/abs/2104.09864>.
- [17] Sidney Black et al. "GPT-NeoX-20B: An Open-Source Autoregressive Language Model". In: *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*. Association for Computational Linguistics, 2022, pp. 95–136. doi: 10.18653/v1/2022.bigsscience-1.9.
- [18] Tian Ye et al. "Physics of Language Models: Part 2.1, Grade-School Math and the Hidden Reasoning Process". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://arxiv.org/abs/2407.20311>.

- [19] Pierre-Carl Langlais et al. *It's All Training: A Fully Synthetic Single-Stage Recipe for LLMs*. 2026. arXiv: 2609.37891 [cs.CL]. URL: <https://arxiv.org/abs/2609.37891>.
- [20] Keller Jordan et al. *modded-nanogpt: Speedrunning the NanoGPT baseline*. 2024. URL: <https://github.com/KellerJordan/modded-nanogpt>.
- [21] Noam Shazeer. *GLU Variants Improve Transformer*. 2020. arXiv: 2002.05202 [cs.LG]. URL: <https://arxiv.org/abs/2002.05202>.
- [22] Stefan Elfving, Eiji Uchibe, and Kenji Doya. *Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning*. 2017. arXiv: 1702.03118 [cs.LG]. URL: <https://arxiv.org/abs/1702.03118>.
- [23] Joshua Ainslie et al. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. arXiv: 2305.13245 [cs.CL]. URL: <https://arxiv.org/abs/2305.13245>.
- [24] Shuangfei Zhai. *Exclusive Self Attention*. 2026. arXiv: 2603.09078 [cs.LG]. URL: <https://arxiv.org/abs/2603.09078>.
- [25] Zihan Qiu et al. *Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free*. 2025. arXiv: 2505.06708 [cs.CL]. URL: <https://arxiv.org/abs/2505.06708>.
- [26] Alex Henry et al. *Query-Key Normalization for Transformers*. 2020. arXiv: 2010.04245 [cs.CL]. URL: <https://arxiv.org/abs/2010.04245>.
- [27] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: <https://arxiv.org/abs/1512.03385>.
- [28] Defa Zhu et al. “Hyper-Connections”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=9FqARW7dwB>.
- [29] Kimi Team et al. *Attention Residuals*. 2026. DOI: 10.48550/arXiv.2603.15031. arXiv: 2603.15031 [cs.CL]. URL: <https://arxiv.org/abs/2603.15031>.
- [30] Zhenda Xie et al. *mHC: Manifold-Constrained Hyper-Connections*. 2026. arXiv: 2512.24880 [cs.CL]. URL: <https://arxiv.org/abs/2512.24880>.
- [31] Da Xiao et al. “MUDDFormer: Breaking Residual Bottlenecks in Transformers via Multiway Dynamic Dense Connections”. In: *Proceedings of the 42nd International Conference on Machine Learning*. Vol. 267. Proceedings of Machine Learning Research. PMLR, 2025, pp. 68440–68458. URL: <https://proceedings.mlr.press/v267/xiao25d.html>.
- [32] Matteo Pagliardini et al. *DenseFormer: Enhancing Information Flow in Transformers via Depth Weighted Averaging*. 2024. arXiv: 2402.02622 [cs.CL]. URL: <https://arxiv.org/abs/2402.02622>.
- [33] NVIDIA Corporation. *NVIDIA H100 Tensor Core GPU Architecture*. Whitepaper. NVIDIA Corporation. 2022. URL: <https://resources.nvidia.com/en-us-hopper-architecture>.
- [34] Zichong Li et al. *NorMuon: Making Muon More Efficient and Scalable*. 2025. arXiv: 2510.05491. URL: <https://arxiv.org/abs/2510.05491>.
- [35] Keller Jordan et al. *Muon: An Optimizer for Hidden Layers in Neural Networks*. 2024. URL: <https://kellerjordan.github.io/posts/muon/>.
- [36] Noah Amsel et al. “The Polar Express: Optimal Matrix Sign Methods and Their Application to the Muon Algorithm”. In: *International Conference on Learning Representations*. 2026, pp. 138323–138360. URL: https://proceedings.iclr.cc/paper_files/paper/2026/hash/df88b275bef31ac96c85f0c4013734fc-Abstract-Conference.html.
- [37] Yu. E. Nesterov. “A method of solving a convex programming problem with convergence rate $O(1/k^2)$ ”. In: *Doklady Akademii Nauk SSSR* 269.3 (1983), pp. 543–547. URL: <https://www.mathnet.ru/eng/dan/v269/i3/p543>.

- [38] Ilya Sutskever et al. “On the Importance of Initialization and Momentum in Deep Learning”. In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. 3. PMLR, 2013, pp. 1139–1147. URL: <https://proceedings.mlr.press/v28/sutskever13.html>.
- [39] Lizhang Chen et al. “Cautious Weight Decay”. In: *International Conference on Learning Representations*. 2026, pp. 126743–126774. URL: https://proceedings.iclr.cc/paper_files/paper/2026/hash/cda57fa0f0ea4c546c39cede9b372173-Abstract-Conference.html.
- [40] Aaron Defazio. *Why Gradients Rapidly Increase Near the End of Training*. 2025. arXiv: 2506.02285 [cs.LG]. URL: <https://arxiv.org/abs/2506.02285>.
- [41] Guangxuan Xiao et al. *SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models*. 2024. arXiv: 2211.10438 [cs.CL]. URL: <https://arxiv.org/abs/2211.10438>.
- [42] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [43] Syeda Nahida Akter et al. “Front-Loading Reasoning: The Synergy between Pretraining and Post-Training Data”. In: *International Conference on Learning Representations*. 2026, pp. 80909–80928. URL: https://proceedings.iclr.cc/paper_files/paper/2026/hash/82eac7050fb44b662062ac64aa6637c3-Abstract-Conference.html.
- [44] Liang Wang et al. *Thinking Augmented Pre-training*. 2025. arXiv: 2509.20186 [cs.CL]. URL: <https://arxiv.org/abs/2509.20186>.
- [45] Ellen Xiaoping Tan et al. *Self-Improving Pretraining: using post-trained models to pretrain better models*. 2026. arXiv: 2601.21343 [cs.CL]. URL: <https://arxiv.org/abs/2601.21343>.
- [46] Jingyan Shen et al. *Understanding Reasoning from Pretraining to Post-Training*. 2026. arXiv: 2607.16097 [cs.LG]. URL: <https://arxiv.org/abs/2607.16097>.
- [47] Mitchell Wortsman et al. “Model Soups: Averaging Weights of Multiple Fine-Tuned Models Improves Accuracy without Increasing Inference Time”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 23965–23998. URL: <https://proceedings.mlr.press/v162/wortsman22a.html>.
- [48] Team OLMo et al. *2 OLMo 2 Furious*. 2025. arXiv: 2501.00656 [cs.CL]. URL: <https://arxiv.org/abs/2501.00656>.
- [49] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531. URL: <https://arxiv.org/abs/1503.02531>.
- [50] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2024. arXiv: 2305.18290 [cs.LG]. URL: <https://arxiv.org/abs/2305.18290>.
- [51] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL]. URL: <https://arxiv.org/abs/2203.02155>.
- [52] Jason Wei et al. *Finetuned Language Models Are Zero-Shot Learners*. 2022. arXiv: 2109.01652 [cs.CL]. URL: <https://arxiv.org/abs/2109.01652>.
- [53] Penghui Qi et al. *Rethinking the Trust Region in LLM Reinforcement Learning*. 2026. arXiv: 2602.04879 [cs.LG]. URL: <https://arxiv.org/abs/2602.04879>.
- [54] Qiying Yu et al. *DAPO: An Open-Source LLM Reinforcement Learning System at Scale*. 2025. arXiv: 2503.14476 [cs.LG]. URL: <https://arxiv.org/abs/2503.14476>.
- [55] Yu Yue et al. *VAPOR: Efficient and Reliable Reinforcement Learning for Advanced Reasoning Tasks*. 2025. arXiv: 2504.05118 [cs.AI]. URL: <https://arxiv.org/abs/2504.05118>.
- [56] Pavel Izmailov et al. “Averaging Weights Leads to Wider Optima and Better Generalization”. In: *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*. 2018, pp. 876–885. URL: <https://www.auai.org/uai2018/proceedings/papers/313.pdf>.

- [57] Kian Ahrabian et al. *A Practical Analysis of Human Alignment with *PO*. 2025. arXiv: 2407.15229 [cs.CL]. URL: <https://arxiv.org/abs/2407.15229>.
- [58] Sadhika Malladi et al. *TailSFT: Filtered Fine-Tuning Improves Post-Training Performance*. 2026. arXiv: 2608.25756 [cs.LG]. URL: <https://arxiv.org/abs/2608.25756>.
- [59] Xingyu Dang et al. *Weight Ensembling Improves Reasoning in Language Models*. 2025. arXiv: 2504.10478 [cs.LG]. URL: <https://arxiv.org/abs/2504.10478>.
- [60] Sen Xu et al. *VibeThinker-3B: Exploring the Frontier of Verifiable Reasoning in Small Language Models*. 2026. arXiv: 2606.16140 [cs.AI]. URL: <https://arxiv.org/abs/2606.16140>.
- [61] Team Olmo et al. *Olmo 3*. 2026. arXiv: 2512.13961 [cs.CL]. URL: <https://arxiv.org/abs/2512.13961>.
- [62] Scott Geng et al. *The Delta Learning Hypothesis: Preference Tuning on Weak Data can Yield Strong Gains*. 2025. arXiv: 2507.06187 [cs.AI]. URL: <https://arxiv.org/abs/2507.06187>.
- [63] David Saxton et al. *Analysing Mathematical Reasoning Abilities of Neural Models*. 2019. arXiv: 1904.01557 [cs.LG]. URL: <https://arxiv.org/abs/1904.01557>.
- [64] Zhihong Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. 2024. arXiv: 2402.03300 [cs.CL]. URL: <https://arxiv.org/abs/2402.03300>.
- [65] Andrei Baroian and Rutger Berger. *Prompt replay: speeding up grpo with on-policy reuse of high-signal prompts*. 2026. arXiv: 2603.21177 [cs.LG]. URL: <https://arxiv.org/abs/2603.21177>.
- [66] Tianshu Zhu et al. *Rollout Pass-Rate Control: Steering Binary-Reward RL Toward Its Most Informative Regime*. 2026. arXiv: 2605.05112 [cs.LG]. URL: <https://arxiv.org/abs/2605.05112>.
- [67] Robin Faro et al. *Frontier Learning: Training LLM Reasoners at the Edge of Capability*. 2026. arXiv: 2609.35426 [cs.LG]. URL: <https://arxiv.org/abs/2609.35426>.
- [68] Yang Yue et al. *Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?* 2025. arXiv: 2504.13837 [cs.AI]. URL: <https://arxiv.org/abs/2504.13837>.
- [69] Xumeng Wen et al. “Reinforcement Learning with Verifiable Rewards Implicitly Incentivizes Correct Reasoning in Base LLMs”. In: *International Conference on Learning Representations*. Ed. by C. Vondrick et al. Vol. 2026. 2026, pp. 49450–49483. URL: https://proceedings.iclr.cc/paper_files/paper/2026/file/517f9b9c227b9dd51dba4560f37165ed-Paper-Conference.pdf.
- [70] Qwen Team. *Qwen3.8-Max: A New Bar for Coding and Cowork*. Aug. 2026. URL: <https://qwen.ai/blog?id=qwen3.8>.
- [71] Qwen Team. *Qwen3.8-Flash-Next: A New Architecture, Towards Ultimate Cost-Efficiency*. Aug. 2026. URL: <https://qwen.ai/blog?id=qwen3.8-flash-next>.
- [72] MiniMax. *MiniMax-M1: Scaling Test-Time Compute Efficiently with Lightning Attention*. 2025. arXiv: 2506.13585 [cs.CL]. URL: <https://arxiv.org/abs/2506.13585>.
- [73] Mika Senghaas et al. “PRIME-RL: Async & Decentralized RL Training at Scale”. In: *NeurIPS 2025 Workshop on GPU-Accelerated and Scalable Optimization*. 2025. URL: <https://openreview.net/pdf?id=yk3ICpEbv8>.
- [74] Michael Noukhovitch et al. “Asynchronous RLHF: Faster and More Efficient Off-Policy RL for Language Models”. In: *International Conference on Learning Representations*. 2025, pp. 4003–4029. URL: https://proceedings.iclr.cc/paper_files/paper/2025/hash/0b99315234cc95e6ef281f9155b68832-Abstract-Conference.html.
- [75] Shengyi Huang et al. “Cleanba: A Reproducible and Efficient Distributed Reinforcement Learning Platform”. In: *International Conference on Learning Representations*. 2024, pp. 6878–6889. URL: https://proceedings.iclr.cc/paper_files/paper/2024/hash/1bd8a2967508c5cf068ac204151b6ecd-Abstract-Conference.html.

- [76] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: [1707.06347](#) [cs.LG]. URL: <https://arxiv.org/abs/1707.06347>.
- [77] Ronald J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Machine Learning* 8 (1992), pp. 229–256. DOI: [10.1007/BF00992696](#).
- [78] Zheng Yuan et al. *Scaling Relationship on Learning Mathematical Reasoning with Large Language Models*. 2023. arXiv: [2308.01825](#). URL: <https://arxiv.org/abs/2308.01825>.
- [79] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2015. arXiv: [1506.02438](#) [cs.LG]. URL: <https://arxiv.org/abs/1506.02438>.
- [80] Zichen Liu et al. *Understanding R1-Zero-Like Training: A Critical Perspective*. 2025. arXiv: [2503.20783](#) [cs.LG]. URL: <https://arxiv.org/abs/2503.20783>.
- [81] Haoxuan Pan, Chuyue Zhou, Xin Li, et al. *JustRL II: Scaling Small LLMs to 128K Reasoning with a Critic*. <https://panhaoxuan.notion.site/justrl-ii-scaling-small-llms-to-128k-reasoning-with-a-critic>. 2026.
- [82] Hynek Kydlíček. *Math-Verify: Math Verification Library*. Version 0.9.0. Software. URL: <https://github.com/huggingface/Math-Verify> (visited on 10/06/2026).

APPENDIX

A NORMUON

Algorithm 1 gives the NorMuon update for one hidden weight matrix in the final pre-training run. Data-parallel communication is omitted: gradients are averaged with a reduce-scatter along the bank dimension, so every rank updates complete matrices, and the result is all-gathered. The coefficients (a_k, b_k, c_k) are the Polar Express schedule for 5 iterations with safety factor $2 \cdot 10^{-2}$ and cushion 2. Steps 2 and 4, in violet, are where the update departs from the original NorMuon [34].

Algorithm 1 NorMuon update for one hidden weight matrix

Input: weight $W \in \mathbb{R}^{m \times n}$ (fp32 master), gradient G_t averaged over ranks, momentum buffer B_{t-1} , second-moment buffer ν_{t-1} (both fp32, $B_0 = 0, \nu_0 = 0$)

Input: momentum μ_t , $\beta_2 = 0.9$, learning rate η_t , weight decay $\lambda = 1.2$, Polar Express coefficients $(a_k, b_k, c_k)_{k=1}^5$

```

1. Nesterov momentum (fp32)
1  $B_t \leftarrow \mu_t B_{t-1} + (1 - \mu_t) G_t$ 
2  $\tilde{G}_t \leftarrow (1 - \mu_t) G_t + \mu_t B_t$ 

2. Polar Express orthogonalisation (bf16, fp32 accumulation)
3  $X_0 \leftarrow \text{bf16}(\tilde{G}_t)$ 
4  $X_0 \leftarrow X_0 / (1.02 \|X_0\|_F + 10^{-6})$   $\triangleright \|X_0\|_2 \leq \|X_0\|_F$ , so spectral norm  $< 1$ 
5 for  $k = 1, \dots, 5$  do
6   if  $m > n$  then
7      $A \leftarrow X_{k-1}^\top X_{k-1}; \quad X_k \leftarrow a_k X_{k-1} + X_{k-1} (b_k A + c_k A^2)$ 
8   else
9      $A \leftarrow X_{k-1} X_{k-1}^\top; \quad X_k \leftarrow a_k X_{k-1} + (b_k A + c_k A^2) X_{k-1}$ 
10  end if
11 end for
12  $U \leftarrow X_5$ 

3. NorMuon variance reduction (statistics in fp32)
13  $r \leftarrow \text{columns if } m \geq n, \text{ otherwise rows}$   $\triangleright$  the longer dimension;  $W_v$  always uses columns
14  $s \leftarrow \text{mean}_r(U \odot U)$   $\triangleright s \in \mathbb{R}^m \text{ or } \mathbb{R}^n; n_r = \text{size of } r$ 
15  $\rho \leftarrow (n_r \sum s)^{1/2}$   $\triangleright = \|U\|_F$ 
16  $\nu_t \leftarrow \beta_2 \nu_{t-1} + (1 - \beta_2) s$ 
17  $\sigma \leftarrow \max(\nu_t, 10^{-10})^{-1/2}$ 
18  $\rho' \leftarrow (n_r \sum s \odot \sigma^2)^{1/2}$   $\triangleright = \|U \odot \sigma\|_F$ 
19  $\Delta \leftarrow U \odot \sigma \cdot \rho / \max(\rho', 10^{-10})$   $\triangleright \sigma$  broadcast along  $r$ ;  $\|\Delta\|_F = \|U\|_F$ 

4. Cautious weight decay and update (fp32 master)
20  $C \leftarrow \mathbb{1}[\Delta \odot W > 0]$   $\triangleright$  strict: no decay where  $\Delta_{ij} = 0$ 
21  $W \leftarrow W - \lambda \eta_t^2 (C \odot W) - \eta_t \Delta$ 
22 return  $W, B_t, \nu_t$ 

```

B REINFORCE, WITHOUT ANY BASELINE, IS ACTUALLY RFT

Let o_i be the response generated by the policy π_θ to a query q_i , then:

$$R_{i,t} = \begin{cases} 1 & \text{if } t = |o_i| \text{ and } o_i \text{ has a correct final answer} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Let B , C and γ be, respectively, the batch, its subset containing only correct rollouts and the discount factor, which controls how much a future reward is valued compared with an immediate one. Then, the policy gradient estimate in the REINFORCE algorithm is:

$$\begin{aligned} \hat{g}_{\text{REINFORCE}} &= \frac{1}{|B|} \sum_{i=1}^{|B|} \left[\sum_{t=1}^{|o_i|} \gamma^t \nabla_\theta \log \pi_\theta(o_{i,t} | q_i, o_{i,<t}) \sum_{\tau=t}^{|o_i|} \gamma^{\tau-t} R_{i,\tau} \right] \\ &= \frac{1}{|B|} \sum_{i=1}^{|B|} \left[\sum_{t=1}^{|o_i|} \nabla_\theta \log \pi_\theta(o_{i,t} | q_i, o_{i,<t}) \sum_{\tau=t}^{|o_i|} R_{i,\tau} \right] \end{aligned} \quad (i)$$

$$= \frac{1}{|B|} \sum_{i=1}^{|B|} R_{i,|o_i|} \sum_{t=1}^{|o_i|} \nabla_\theta \log \pi_\theta(o_{i,t} | q_i, o_{i,<t}) \quad (ii)$$

$$= \frac{1}{|B|} \sum_{i \in C} \sum_{t=1}^{|o_i|} \nabla_\theta \log \pi_\theta(o_{i,t} | q_i, o_{i,<t}) \quad (iii)$$

The equalities follow from:

- (i) Setting $\gamma = 1$
- (ii) Assigning zero reward to each state different from the last one
- (iii) Assigning $R_{i,|o_i|} = 0$ to wrong rollouts and $R_{i,|o_i|} = 1$ to correct ones

As you can see, under these settings, REINFORCE and online RFT have the same update direction. Indeed, the policy gradient is proportional to the negative gradient of the online RFT loss, which is the cross entropy loss on correct solutions sampled from π_θ .

Formally:

$$\mathcal{L}_{\text{RFT}}(\theta) = -\frac{1}{|C|} \sum_{i \in C} \sum_{t=1}^{|o_i|} \log \pi_\theta(o_{i,t} | q_i, o_{i,<t}) \quad (7)$$

$$\hat{g}_{\text{RFT}} = -\frac{1}{|C|} \sum_{i \in C} \sum_{t=1}^{|o_i|} \nabla_\theta \log \pi_\theta(o_{i,t} | q_i, o_{i,<t}) \quad (8)$$

Therefore:

$$\hat{g}_{\text{REINFORCE}} \propto -\hat{g}_{\text{RFT}}$$

C VALUE FUNCTIONS AND ADVANTAGES DEFINITIONS

- $Q^\pi(s_t, a_t)$ is the expected return of taking the action a_t in the state s_t and then following policy π .
- $V^\pi(s_t)$ is the expected return from the state s_t under policy π .
- $A^\pi(s_t, a_t)$ is the advantage of taking the action a_t in the state s_t for the policy π

Namely:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{\tau=0}^{\infty} \gamma^\tau R_{i,t+\tau} \mid s_t = s, a_t = a \right] \quad (9)$$

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)] \quad (10)$$

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \quad (11)$$

In our case, $\gamma = 1$ and rewards are defined as in Equation (6).

Let s be $[q, o_{<t}]$, where q is a query and o is the rollout sampled by π as a response to q , then under our settings:

$$Q^\pi(s, o_t) = \mathbb{E}_\pi [R_{|o|} \mid s_t = s, a_t = o_t] \quad (12)$$

$$V^\pi(s) = \mathbb{E}_\pi [R_{|o|} \mid s_t = s] \quad (13)$$

$$A^\pi(s, o_t) = Q^\pi(s, o_t) - V^\pi(s) \quad (14)$$

D GAE

GAE allows us to control the bias-variance trade-off through $\lambda \in [0, 1]$: lower values imply more bootstrapping from the value model, while $\lambda = 1$ uses an unbiased Monte Carlo estimate of the return.

In detail, given the timestep $t = 1, \dots, T$, set the value after the final timestep equal to 0: $V_\phi^\pi(s_{T+1}) = 0$, then:

$$\delta_{i,t} = R_{i,t} + \gamma V_\phi^\pi(s_{t+1}) - V_\phi^\pi(s_t) \quad (15)$$

$$\hat{A}_{i,t}^{\text{GAE}} = \sum_{\tau=0}^{T-t} (\gamma\lambda)^\tau \delta_{i,t+\tau} \quad (16)$$

In our case, $T = |o_i|$ which is the length of the model rollout o_i , $\gamma = 1$, and rewards are defined as in Equation (6).

Therefore:

$$\delta_{i,t} = \begin{cases} R_{i,|o_i|} - V_\phi^\pi(s_{|o_i|}) & \text{if } t = |o_i| \\ V_\phi^\pi(s_{t+1}) - V_\phi^\pi(s_t) & \text{otherwise} \end{cases} \quad (17)$$

$$\hat{A}_{i,t}^{\text{GAE}} = \sum_{\tau=0}^{|o_i|-t} \lambda^\tau \delta_{i,t+\tau} \quad (18)$$

$$\hat{A}_{i,t}^{\text{GAE}} = \sum_{\tau=0}^{|o_i|-t} \delta_{i,t+\tau} = R_{i,|o_i|} - V_\phi^\pi(s_t) \quad \text{if } \lambda = 1 \quad (19)$$

E LENGTH-NORMALIZED DPO OBJECTIVE

Let π_θ be the trained policy and π_{ref} the frozen reference policy. Both are initialized from the distilled soup. Each training example is a query q with a preferred response o^+ and a dispreferred response o^- . Standard DPO [50] defines the implicit reward as the *summed* log-ratio over the response tokens. We instead use its *mean* over the response length [57]:

$$r_\theta(q, o) = \frac{1}{|o|} \sum_{t=1}^{|o|} \log \frac{\pi_\theta(o_t \mid q, o_{<t})}{\pi_{\text{ref}}(o_t \mid q, o_{<t})}. \quad (20)$$

The loss over the preference dataset $\mathcal{D}$ is

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(q, o^+, o^-) \sim \mathcal{D}} \left[\log \sigma \left(\beta (r_\theta(q, o^+) - r_\theta(q, o^-)) \right) \right], \quad (21)$$

with $\beta = 5$. With the summed reward, the margin grows with the number of tokens. A pair whose responses differ in length therefore gets a reward gap that partly reflects length rather than quality. Dividing by $|o|$ puts the rewards of both responses on a per-token scale, which removes this bias.

F COMPUTE ESTIMATES FOR FIGURE 1

Training compute in Figure 1 is the analytical $6ND$ estimate, with N the parameter count (active parameters for mixture-of-experts models) and D the number of training tokens. It is an estimate, not a measured total. For Limite 1B - Violetto it counts pretraining, offline distillation and SFT, and excludes RL. For the other models it comes from public reports and model cards, and the training stages counted differ by model.

The marked points need the following qualifications.

- *VibeThinker 3B: the Coder training stage only.
- †MiniCPM5 2B: the 1T midtraining tokens only. The solid arrow marks the earlier pretraining this leaves out.
- ‡Qwen3.5: assumes the 36T tokens reported for Qwen3. The dashed arrows mark larger budgets implied by family-level claims; they are not verified bounds.
- §Qwen3.5 4B: the score is IRT-estimated.

Scores for Qwen3.5 and Nemotron 3 Super are from MathArena, scores for Nemotron 3 Nano and Nemotron Cascade 2 are from NVIDIA, and the rest are our own evaluations. Evaluation setups differ across these sources.

G DETAILED RESULTS BY CONTEXT LENGTH

Table 8: Benchmark accuracy (%) across post-training stages at a 64k context length. Gains are measured in percentage points relative to the immediately preceding checkpoint at the same context length. [†] SFT post-RL scores and gains are offline estimates from rollouts generated at 128k: responses were retokenized, truncated to 61,440 tokens, and evaluated with the same scorers. No new rollouts were generated at 64k. Original token sequences were not saved, and generation with a different limit may produce different responses. DPO and RL evaluations also used different concurrency settings between the 64k and 128k runs.

Benchmark	Off-policy distillation	DPO pre-RL		First RL		Second RL		SFT post-RL [†]	
	Score	Score	Gain	Score	Gain	Score	Gain	Score	Gain
AIME 2025	77.08	80.52	+3.44	85.62	+5.10	87.71	+2.09	≈ 87.40	≈ -0.31
AIME 2026	81.67	83.54	+1.87	87.50	+3.96	87.40	-0.10	≈ 90.42	≈ +3.02
HMMT Feb. 2025	70.42	76.15	+5.73	80.73	+4.58	80.62	-0.11	≈ 85.94	≈ +5.32
HMMT Feb. 2026	64.68	62.41	-2.27	73.01	+10.60	72.25	-0.76	≈ 75.66	≈ +3.41
APEX Shortlist	23.27	21.81	-1.46	33.44	+11.63	32.05	-1.39	≈ 38.16	≈ +6.11
BeyondAIME	57.16	52.25	-4.91	65.28	+13.03	65.34	+0.06	≈ 67.78	≈ +2.44
ArXivMath May 2026	20.08	15.39	-4.69	25.08	+9.69	24.14	-0.94	≈ 21.09	≈ -3.05

Table 9: Benchmark accuracy (%) across post-training stages at a 128k context length. Gains are measured in percentage points relative to the immediately preceding checkpoint at the same context length. DPO and RL evaluations also used different concurrency settings between the 64k and 128k runs.

Benchmark	Off-policy distillation	DPO pre-RL		First RL		Second RL		SFT post-RL	
	Score	Score	Gain	Score	Gain	Score	Gain	Score	Gain
AIME 2025	79.58	87.29	+7.71	85.42	-1.87	87.40	+1.98	89.17	+1.77
AIME 2026	81.77	91.25	+9.48	88.54	-2.71	86.98	-1.56	91.35	+4.37
HMMT Feb. 2025	72.40	81.77	+9.37	81.98	+0.21	79.48	-2.50	89.69	+10.21
HMMT Feb. 2026	66.48	74.15	+7.67	72.16	-1.99	72.16	0.00	80.78	+8.62
APEX Shortlist	25.33	34.51	+9.18	35.17	+0.66	30.12	-5.05	43.42	+13.30
BeyondAIME	59.41	66.03	+6.62	66.03	0.00	65.34	-0.69	72.38	+7.04
ArXivMath May 2026	20.47	24.69	+4.22	26.09	+1.40	25.86	-0.23	23.52	-2.34

Table 10: Standard errors (percentage points) of the pass@1 scores in Table 7. For each of the $R = 32$ generation repetitions, we compute accuracy over the full benchmark; the SE is the sample standard deviation of these accuracies divided by $\sqrt{R}$. [†] denotes externally reported scores, for which no SE is available.

Model	AIME 2026	HMMT Feb. 2026	APEX Shortlist	Beyond AIME	AIME 2025	HMMT Feb. 2025	ArXivMath May 2026
Limite 1B - Violetto	0.54	0.74	0.88	1.03	0.49	0.72	0.81
VibeThinker-1.5B	0.95	0.81	0.56	0.37	1.08	1.20	0.62
MiniCPM5-2B	0.65	0.95	0.99	0.53	0.75	0.92	0.56
VibeThinker-3B	0.52	0.72	1.24	0.40	0.53	0.79	0.91
Qwen3.5-4B	[†]	[†]	[†]	1.11	0.87	1.11	0.51
Olmo-3-7B-Think	1.17	0.70	0.54	0.40	0.98	0.89	0.56
Olmo-3-7B-Instruct	1.18	0.59	0.49	0.54	1.00	0.91	0.57
OLMo-Hybrid-Think-SFT-7B	0.86	0.71	0.57	0.46	1.07	0.91	0.46
Qwen3.5-9B	0.61	0.74	0.79	0.46	0.76	0.72	0.73
MUSE-Glimmer-30B	0.44	0.64	1.00	0.82	0.46	0.72	0.74
Gemma-4-31B-IT	0.52	0.56	0.85	1.78	0.81	0.78	0.55

H QUALITATIVE EXAMPLES

The following selected, independent single-turn interactions illustrate both Limite's mathematical reasoning and a limitation in instruction following. They are illustrative examples, not an estimate of how often either behavior occurs. The original exchanges appear in the release post.²

H.1 LIMITATION IN INSTRUCTION FOLLOWING

Example 1: Photosynthesis

User:

What is photosynthesis, in simple terms?

Limite:

A cell divides into two cells, and those cells can each divide into two...

H.2 A HARD PROBLEM, SOLVED

Example 2: A path through a board

User:

Consider a 2025×2025 board whose squares are identified with pairs (i, j) , where i and j denote the row and column.

Calvin picks two positive integers $a, b < 2025$ and places a pawn at $(1, 1)$. On odd-numbered moves, he moves from (i, j) to either $(i + a, j)$ or $(i, j + a)$. On even-numbered moves, he moves to either $(i + b, j)$ or $(i, j + b)$. All coordinates are taken modulo 2025.

For how many pairs (a, b) can the pawn visit every square exactly once?

Limite:

Thus a necessary condition is

$$a = b \quad \text{and} \quad \gcd(a, 2025) = 1.$$

[...]

Write $t = kn + s$ with $0 \leq k, s \leq n - 1$. Then $R_t = k$ and

$$y_t \equiv kn + s - k \equiv s - k \pmod{n}.$$

Thus the visited vertex is $(k, s - k \pmod{n})$. As (k, s) runs over all n^2 pairs, $(k, s - k)$ also runs over all n^2 pairs, so the path visits every square exactly once.

[...]

The admissible pairs are precisely (a, a) with $1 \leq a \leq n - 1$ and $\gcd(a, n) = 1$. The number of such a is Euler's totient $\varphi(n)$.

$$\varphi(2025) = 2025 \left(1 - \frac{1}{3}\right) \left(1 - \frac{1}{5}\right) = 1080.$$

1080

²<https://paradigma.inc/blog/limite-1b-violetto/>

ACKNOWLEDGEMENTS

We thank Donato Crisostomi for supervision and project management early in the project, Alessandro Fazio for support with Flywheel development, Alessandro Lombardo for operational support, and all three of them, along with Francesco Prignano, for the many exchanges of ideas.

We thank Tony Silveti-Falls and Minh Nhat Nguyen for feedback on a pre-release draft of this report.

We also thank Manfredi Caracciolo for the aura artwork.