Reinforcement Learning Outperforms Supervised Fine-Tuning for Mathematical Reasoning in Small Language Models: Ablation Studies on GRPO, LoRA, and Generation Context

Howard Cheng
cheng.howard1@gmail.com

March 2026

Abstract

We present a systematic study of post-training strategies for improving mathematical reasoning in a 4-billion parameter language model. Starting from Qwen3-4B-Instruct, we compare supervised fine-tuning (SFT) via teacher distillation against reinforcement learning (RL) using Group Relative Policy Optimization (GRPO) with verifiable math rewards. Our experiments yield several key findings: (1) GRPO on hard problems achieves 90.2% on MATH-500, a +10.8 percentage point improvement over the 79.4% baseline, while SFT *degrades* performance by −2.2pp; (2) SFT warm-starting provides no benefit over direct RL; (3) training exclusively on hard problems (MATH Level 4–5) with extended generation context (4096 tokens) is the single most impactful intervention; (4) LoRA fine-tuning with optimal hyperparameters outperforms full-parameter training on a single GPU, achieving comparable accuracy in 50 steps versus 998; and (5) inference-time majority voting (maj@16) further boosts MATH-500 accuracy to 93.2% and AIME 2024 accuracy to 56.7%. Our ablation studies isolate the contributions of training data difficulty, generation context length, group size, parameter efficiency (LoRA vs. full-parameter), and continued training, providing practical guidelines for post-training small language models for mathematical reasoning.

1 Introduction

Large language models (LLMs) have demonstrated remarkable mathematical reasoning capabilities, yet smaller models ($\leq$ 4B parameters) remain significantly behind their larger counterparts on challenging benchmarks such as AIME competition problems. Two dominant post-training paradigms have emerged for improving mathematical reasoning: supervised fine-tuning (SFT) on expert reasoning traces, and reinforcement learning (RL) with verifiable rewards.

Recent work has shown that RL with outcome-based rewards—particularly Group Relative Policy Optimization (GRPO) [Shao et al., 2024]—can substantially improve mathematical reasoning without requiring process-level reward models. Meanwhile, teacher distillation via SFT remains a common first step in many training pipelines [Mukherjee et al., 2023, Yu et al., 2024]. However, the relative effectiveness of these approaches for *small* models, and whether SFT provides a useful warm-start for subsequent RL, remains underexplored.

In this work, we conduct a comprehensive ablation study using Qwen3-4B-Instruct (4B parameters) as our base model, training on competition-level math problems from the MATH dataset [Hendrycks et al., 2021]. Our contributions include:

1. A systematic comparison of SFT, direct RL, and SFT→RL pipelines, demonstrating that SFT is counterproductive for small models with existing math capabilities (Section 5.2).

2. An ablation of GRPO training data composition and generation context length, showing that training on hard problems only with 4096-token completions yields a +5.0pp improvement over training on all difficulties with 2048 tokens (Section 5.3).

3. A controlled comparison of LoRA versus full-parameter GRPO on identical hardware (H100 80GB), revealing that LoRA achieves equivalent accuracy in 50× fewer training steps under single-GPU constraints (Section 5.5).

4. An inference-time scaling analysis using majority voting, identifying maj@16 as the optimal budget and characterizing the gap between majority voting and oracle (pass@k) performance (Section 5.6).

2 Related Work

Mathematical Reasoning in LLMs. Mathematical reasoning has emerged as a key benchmark for LLM capability. The MATH dataset [Hendrycks et al., 2021] and GSM8K [Cobbe et al., 2021] have become standard evaluation benchmarks, while competition-level benchmarks such as AIME provide ceiling tests. Recent models including DeepSeek-R1 [DeepSeek-AI, 2025] and QwQ [QwQ Team, 2024] have achieved strong performance through extended reasoning chains, often trained via RL.

Reinforcement Learning for Reasoning. GRPO [Shao et al., 2024] eliminates the need for a separate critic model by using group-relative advantages: for each problem, K completions are sampled and advantages are computed relative to the group mean reward. This approach has been scaled in DeepSeek-R1 [DeepSeek-AI, 2025] and adopted by the open-source TRL library [von Werra et al., 2022]. Concurrent work on RLVR (RL with Verifiable Rewards) has shown that binary correctness rewards are sufficient for mathematical domains [Lambert et al., 2024].

Parameter-Efficient Fine-Tuning. LoRA [Hu et al., 2022] and its variants enable training large models with a fraction of the parameter updates. While LoRA is widely used for SFT, its application to RL training and the comparison with full-parameter RL on identical hardware has received less attention.

Teacher Distillation. Knowledge distillation from larger to smaller models [Hinton et al., 2015] is a common strategy for improving small model capabilities. In the mathematical domain, generating reasoning traces from teacher models and fine-tuning student models on these traces has shown mixed results depending on the compatibility between teacher and student reasoning styles [Yu et al., 2024, Mukherjee et al., 2023].

Inference-Time Scaling. Majority voting [Wang et al., 2023] and best-of-n sampling [Cobbe et al., 2021] provide inference-time compute scaling. Recent work has explored the diminishing returns of simple majority voting and the potential of learned verifiers to close the gap between pass@k and majority vote accuracy [Lightman et al., 2024].

3 Experimental Setup

3.1 Models

We use **Qwen3-4B-Instruct-2507** as our base student model (4B parameters). For teacher distillation, we use **Qwen3-235B-A22B-Instruct-2507** (235B parameters, mixture-of-experts with 22B active). Both models are instruction-tuned variants from the Qwen3 family.

3.2 Training Frameworks

We employ two training frameworks:

- **Tinker** [Tinker, 2025]: A cloud-based LoRA fine-tuning API with dedicated inference and training infrastructure. Tinker's architecture separates generation (on an inference cluster) from gradient updates (on training GPUs), enabling larger group sizes and longer generation contexts without GPU memory conflicts.
- **TRL + vLLM**: The open-source TRL library [von Werra et al., 2022] with vLLM [Kwon et al., 2023] in colocate mode for local training on a single NVIDIA H100 80GB GPU. In colocate mode, vLLM shares GPU memory with PyTorch training, using sleep mode to offload weights during the backward pass.

3.3 Training Data

All RL experiments train on problems from the MATH dataset [Hendrycks et al., 2021]:

- **MATH (full):** 7,500 problems across difficulty Levels 1–5, spanning algebra, number theory, geometry, combinatorics, and more.
- **MATH (hard):** 3,994 problems filtered to Levels 4–5 only. These problems require multi-step reasoning and are where the model has the most room for improvement.

For SFT, we generate reasoning traces by prompting the teacher model (Qwen3-235B-A22B-Instruct) on MATH Level 4–5 problems and 500 OlympiadBench [He et al., 2024] numerical problems, filtering for correctness. This yields 3,825 verified traces (3,435 from MATH + 390 from OlympiadBench) with a median length of ∼2,300 characters.

3.4 Reward Function

All RL experiments use a binary correctness reward:

$$r(y, y^*) = \begin{cases} 1 & \text{if } \texttt{grade}(\texttt{extract\_boxed}(y),\ y^*) = \texttt{True} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where y is the model completion, y^* is the ground-truth answer, $\texttt{extract\_boxed}(\cdot)$ parses the final `\boxed{}` expression, and $\texttt{grade}(\cdot, \cdot)$ performs robust mathematical equivalence checking (handling fractions, decimals, simplified forms, and LaTeX formatting).

3.5 Evaluation Protocol

We evaluate on five benchmarks spanning a range of difficulty:

Table 1: Evaluation benchmarks.

Benchmark	Size	Difficulty	Purpose
MATH-500	500	Competition math	Primary metric
AIME 2024	30	Very hard	Ceiling test
AIME 2025	30	Very hard	Temporal OOD
GSM8K	1,319	Grade school	Floor / regression test
OlympiadBench	572	Olympiad-level	Hard benchmark

All evaluations use near-deterministic decoding (temperature = 0.01), a standardized few-shot prefix, and the `\boxed{}` answer format. Confidence intervals are 95% using the normal approximation: $\text{CI} = 1.96\sqrt{p(1-p)/n}$.

4 Methodology

4.1 Supervised Fine-Tuning (SFT)

We fine-tune Qwen3-4B-Instruct on teacher-distilled reasoning traces using LoRA [Hu et al., 2022] with the following configuration:

Table 2: SFT hyperparameters.

Parameter	Value	Notes
LoRA rank	64	Applied to all attention + MLP projections
Learning rate	2×10^{-4}	Linear decay to 0
Batch size	32	119 steps/epoch
Epochs	2	238 total steps
Max sequence length	4,096	
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.95$)	
Loss masking	Assistant tokens only	

We explored six SFT configurations varying data sources (OlympiadBench only, combined custom traces, OpenR1-Math-220k), learning rates (2×10^{-5} to 5×10^{-4}), and epochs (1–3). The best configuration ("Config A") achieves 77.2% on MATH-500, still below the 79.4% baseline. See Section 6 for the full sweep.

4.2 Group Relative Policy Optimization (GRPO)

GRPO [Shao et al., 2024] trains the policy π_θ by generating a group of K completions $\{y_1, \ldots, y_K\}$ for each prompt x, computing rewards $\{r_1, \ldots, r_K\}$, and updating the policy to increase the likelihood of high-reward completions relative to the group mean.

The advantage for each completion is:

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon} \tag{2}$$

where μ_r and σ_r are the mean and standard deviation of rewards within the group. When $\sigma_r = 0$ (all completions correct or all incorrect), the problem contributes zero gradient and is effectively skipped.

The GRPO objective with importance sampling is:

$$\mathcal{L}(\theta) = -\mathbb{E}_{x,\{y_i\}}\left[\sum_{i=1}^{K} \min\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}\hat{A}_i,\ \text{clip}\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}, 1 \pm \epsilon\right)\hat{A}_i\right)\right] \tag{3}$$

We set the KL penalty coefficient $\beta = 0$, as we found no benefit from constraining the policy to stay close to the reference model in our setting. This is consistent with findings from DeepSeek-R1 [DeepSeek-AI, 2025].

4.3 Training Configurations

We run GRPO under multiple configurations to isolate the effect of each design choice:

Table 3: GRPO training configurations. All use LoRA rank=64 with binary correctness reward.

Config	Data	LR	K	Max Tokens	Epochs	Framework
GRPO v1	All MATH	4×10^{-5}	16	2,048	$\sim$1	Tinker
GRPO v2	Hard only	4×10^{-5}	16	4,096	2	Tinker
GRPO v3	Hard only	2×10^{-5}	16	4,096	+2	Tinker
Local (full)	Hard only	5×10^{-6}	8	2,048	2	TRL+vLLM
Local (LoRA)	Hard only	4×10^{-5}	8	2,048	$\sim$0.1	TRL+vLLM

5 Results

5.1 Main Results

Table 4 presents our main ablation results across all five benchmarks. GRPO v2 (LoRA, hard problems, 4096-token context) achieves the best overall performance, improving MATH-500 accuracy from 79.4% to 90.2% (+10.8pp) and AIME 2024 from 20.0% to 43.3% (+23.3pp).

Table 4: Main results across five benchmarks. All evaluations use temperature = 0.01 with greedy decoding. Best results in **bold**. ± values are 95% confidence intervals.

Model	MATH-500	AIME 2024	AIME 2025	GSM8K	OlympiadBench
Baseline	79.4±3.5	20.0±14.3	20.0±14.3	94.1±1.3	49.0±4.1
SFT (custom traces)	77.2±3.7	10.0±10.7	16.7±13.3	93.2±1.4	48.4±4.1
Direct RL (v1)	85.2±3.1	30.0±16.4	26.7±15.8	93.5±1.3	55.8±4.1
SFT → RL	85.4±3.1	23.3±15.1	20.0±14.3	93.4±1.3	54.2±4.1
GRPO v2 (hard)	**90.2**±2.6	**43.3**±17.7	26.7±15.8	93.8±1.3	**63.6**±3.9
GRPO v3 (continued)	90.6±2.6	36.7±17.2	**30.0**±16.4	**94.0**±1.3	64.0±3.9
Local full-param	87.8±2.9	26.7±15.8	26.7±15.8	94.4±1.2	58.9±4.0
Local LoRA (step 50)	87.2±2.9	30.0±16.4	26.7±15.8	93.9±1.3	60.0±4.0

5.2 SFT vs. Reinforcement Learning

Our first key finding is that **SFT degrades performance** while **RL substantially improves it**. Table 5 isolates this comparison:

Table 5: SFT vs. RL comparison. Δ is change from baseline.

	MATH-500	Δ	AIME 2024	Δ
Baseline	79.4%	—	20.0%	—
SFT (best config)	77.2%	−2.2pp	10.0%	−10.0pp
Direct RL (GRPO v1)	85.2%	+5.8pp	30.0%	+10.0pp
GRPO v2 (hard + 4096)	90.2%	+10.8pp	43.3%	+23.3pp

The SFT model performs *worse* than baseline across all benchmarks, including a catastrophic −10.0pp drop on AIME 2024. We hypothesize that at the 4B parameter scale, the model has limited capacity to both maintain its pre-existing mathematical capabilities and learn from distilled traces. The SFT objective pushes the model to mimic the teacher's reasoning style, which may not be optimal for the student's architecture and pre-training distribution.

SFT warm-start provides no benefit. Comparing Direct RL (85.2% MATH-500) versus SFT→RL (85.4% MATH-500), we find no statistically significant difference. The SFT warm-start neither helps nor hurts the subsequent RL training. This is consistent with recent findings that GRPO can discover effective reasoning strategies directly from the base model [DeepSeek-AI, 2025].

5.3 GRPO Ablation: Data Difficulty and Context Length

Our second ablation isolates the effects of training data composition and generation context length. GRPO v1 trains on all MATH problems with 2048-token completions; GRPO v2 restricts to hard problems (Level 4–5) and doubles the context to 4096 tokens.

Table 6: Effect of training data difficulty and generation context length.

Config	Data	Max Tokens	MATH-500	AIME 2024	OlympiadBench
GRPO v1	All levels	2,048	85.2%	30.0%	55.8%
GRPO v2	Level 4–5	4,096	90.2%	43.3%	63.6%
Δ			+5.0pp	+13.3pp	+7.8pp

The improvement is dramatic: +5.0pp on MATH-500, +13.3pp on AIME 2024, and +7.8pp on OlympiadBench. We attribute this to two complementary mechanisms:

Hard problems provide more informative signal. Easy problems (Level 1–3) are solved correctly by most or all of the $K = 16$ completions, yielding zero-variance reward groups ($\sigma_r = 0$) that contribute no gradient. Training exclusively on hard problems ensures that each batch contains informative signal—problems where some completions succeed and others fail.

Longer context enables complete reasoning chains. Competition-level problems often require multi-step derivations spanning hundreds of tokens. With a 2048-token limit, solutions to the hardest problems are truncated mid-reasoning, preventing the model from learning complete solution strategies. Doubling to 4096 tokens allows the model to express and be rewarded for full solutions.

5.4 Effect of Continued Training

GRPO v3 continues training from the v2 checkpoint for 2 additional epochs at half the learning rate (2×10^{-5}). The results show minimal improvement:

Table 7: Continued training (v3) vs. initial training (v2).

	MATH-500	AIME 2024	AIME 2025	OlympiadBench
GRPO v2	90.2%	**43.3%**	26.7%	63.6%
GRPO v3	**90.6%**	36.7%	**30.0%**	**64.0%**
Δ	+0.4pp	−6.6pp	+3.3pp	+0.4pp

MATH-500 and OlympiadBench show marginal gains (+0.4pp each), while AIME 2024 actually *decreases* by 6.6pp (−2 problems out of 30). The mixed results on AIME benchmarks are within the 95% confidence interval (±16–18pp for $n = 30$), suggesting the model is near its single-sample performance ceiling at this scale. Further training produces diminishing—and potentially negative—returns.

5.5 LoRA vs. Full-Parameter Training

To test whether LoRA's low-rank constraint limits GRPO performance, we compare LoRA and full-parameter training on identical hardware (NVIDIA H100 80GB) using TRL's GRPOTrainer with vLLM in colocate mode.

Hardware constraints necessitate hyperparameter compromises. Full-parameter training of 4B parameters requires ∼32 GB for Adam optimizer states alone (fp32 master weights + first and second moments), leaving limited headroom for vLLM generation buffers. We therefore reduce the group size ($K = 8$ vs. 16) and maximum completion length (2048 vs. 4096 tokens) compared to the cloud-based LoRA experiments.

Table 8: LoRA vs. full-parameter GRPO on H100 80GB. Both use identical generation budget ($K = 8$, max tokens = 2048) and training data (MATH Level 4–5).

Method	Steps	Time	MATH-500	AIME 2024	GSM8K	OlympiadBench
Full-param	998	13h	87.8%	26.7%	**94.4%**	58.9%
LoRA (step 50)	50	∼50min	87.2%	**30.0%**	93.9%	**60.0%**
LoRA (step 100)	100	∼100min	86.0%	33.3%	—	—

LoRA matches full-parameter accuracy in 50× fewer steps. Despite training only ∼160M parameters (4% of the total), LoRA at step 50 achieves accuracy comparable to the full-parameter model trained for 998 steps. LoRA actually outperforms full-parameter training on AIME 2024 (+3.3pp) and OlympiadBench (+1.1pp).

LoRA saturates and overfits quickly at high learning rates. At LR $= 4 \times 10^{-5}$, the LoRA model reaches reward ≈ 0.95 by step 50, with gradient norms dropping to $\sim 10^{-7}$ and the importance sampling ratio falling to ∼0.1 (indicating significant policy divergence from the reference). Step 100 shows degradation on MATH-500 (86.0% vs. 87.2% at step 50), confirming overfitting. This suggests that LoRA RL requires careful early stopping or lower learning rates for optimal results.

Generation budget dominates over parameterization. Both local methods (full-param and LoRA) achieve ∼87–88% on MATH-500, well below the Tinker LoRA result of 90.2% despite using the same LoRA rank and training data. The key difference is the generation budget: local runs use $K = 8$ and 2048-token completions, while the Tinker setup uses $K = 16$ and 4096 tokens. This confirms our finding from Section 5.3 that **generation context length and group size are more important than the choice of LoRA vs. full-parameter training**.

Table 9: Impact of generation budget: cloud LoRA (large budget) vs. local methods (small budget).

Config	Method	K	Max Tokens	MATH-500	AIME 2024
Tinker (v2)	LoRA	16	4,096	**90.2%**	**43.3%**
Local	Full-param	8	2,048	87.8%	26.7%
Local	LoRA	8	2,048	87.2%	30.0%

GPU memory analysis. Table 10 shows the estimated memory breakdown explaining why full-parameter training requires a smaller generation budget.

Table 10: GPU memory budget (H100 80GB) for GRPO training.

Component	**Full-Param**	**LoRA (rank=64)**
Model weights (bf16)	8 GB	8 GB
Optimizer states (fp32)	32 GB	0.5 GB
Gradients (bf16)	8 GB	0.3 GB
Activations (w/ grad ckpt)	∼10 GB	∼10 GB
vLLM generation buffer	∼8 GB	∼8 GB
Total peak	∼66 GB	∼27 GB
Headroom for K, context	14 GB	**53 GB**

LoRA's 53 GB of headroom could in principle support the full Tinker configuration ($K = 16$, 4096 tokens), though vLLM's colocate mode and weight-reloading overhead make this impractical on a single GPU from a wall-clock perspective (estimated >60 hours).

5.6 Majority Voting (Inference-Time Scaling)

We apply majority voting to our best model (GRPO v2) by sampling K completions at temperature = 0.7, extracting `\boxed{}` answers, and selecting the most common answer. Mathematically equivalent answers are grouped using the same `grade_answer()` function used during training.

Table 11: Majority voting results for GRPO v2 at temperature = 0.7.

Benchmark	**Greedy**	**maj@16**	**maj@64**	**pass@16**	**pass@64**
MATH-500	90.2%	**93.2%**	93.2%	95.6%	97.2%
AIME 2024	43.3%	**56.7%**	53.3%	66.7%	66.7%
AIME 2025	26.7%	**43.3%**	40.0%	53.3%	53.3%

maj@16 is the optimal voting budget. On all three benchmarks, maj@16 matches or outperforms maj@64. On AIME 2024, maj@64 actually *decreases* from 56.7% to 53.3%. We attribute this to the diversity effect: at higher sample counts with temperature 0.7, the model generates a wider variety of *incorrect* answers that can collectively outvote the correct one. This is particularly pronounced on hard problems where the per-sample success rate is low (∼40–50%).

The gap between maj@k and pass@k reveals the verifier opportunity. On MATH-500, pass@64 = 97.2% (486/500 problems solvable) while maj@64 = 93.2%, a gap of 4.0pp representing 20 problems that the model can solve *sometimes* but that majority voting fails to identify. Closing this gap would require a learned verifier that can reliably select the correct completion from a set containing both correct and incorrect solutions.

6 SFT Experiments Detail

Table 12 presents the full SFT hyperparameter sweep, demonstrating that no configuration improves over the baseline.

Table 12: SFT hyperparameter sweep. All configurations use LoRA rank=64.

Version	Data	LR	Epochs	Steps	MATH-500
Baseline	—	—	—	—	79.4%
v1	390 OlympiadBench	2×10^{-5}	3	9	∼79%
v2	390 OlympiadBench	5×10^{-4}	3	36	∼79%
v3	5.4k (custom + OpenR1)	5×10^{-4}	2	168	∼70%
v4	861 (custom short)	5×10^{-4}	2	53	∼77%
Config A	3,825 (full custom)	2×10^{-4}	2	238	77.2%
Config B	3,825 (full custom)	5×10^{-4}	1	119	76.8%

OpenR1-Math-220k traces cause catastrophic regression. Version v3, which includes traces from OpenR1-Math-220k [OpenR1, 2025], drops MATH-500 to ∼70%, a −9.4pp regression from baseline. The OpenR1 traces are generated by DeepSeek-R1 and have a median length of ∼7,300 characters—3× longer than our custom traces. The reasoning style mismatch between DeepSeek R1's extended thinking pattern and Qwen3's more concise reasoning appears to be the primary cause. Even length-filtering OpenR1 traces to <4,000 characters retains only 9.4% of the data and still underperforms.

Interpretation. For a 4B model with pre-existing mathematical capability, SFT on distilled traces does not teach new reasoning skills; instead, it overwrites the model's learned reasoning distribution with the teacher's style, which may be suboptimal for the student's capacity and architecture. This finding may not hold for models without strong pre-existing math abilities, or for larger models with more capacity to absorb new reasoning patterns without forgetting.

7 Analysis and Discussion

7.1 Why RL Succeeds Where SFT Fails

The divergent outcomes of SFT and RL for mathematical reasoning at the 4B scale can be understood through the lens of what each objective optimizes:

- **SFT** minimizes $D_{\text{KL}}(\pi_{\text{teacher}} \| \pi_\theta)$, forcing the student to match the teacher's *distribution* over reasoning traces. For a small model, this is an overspecified objective that can destroy useful pre-existing capabilities.
- **RL (GRPO)** maximizes $\mathbb{E}_{y \sim \pi_\theta}[r(y)]$, allowing the model to discover its *own* effective reasoning strategies within its capacity constraints. The model is free to use shorter, more direct reasoning paths that would receive low likelihood under the teacher distribution but achieve the correct answer.

This asymmetry is particularly pronounced for small models where capacity is limited. The model cannot simultaneously maintain its original capabilities and faithfully reproduce long-form teacher reasoning traces.

7.2 The Generation Budget Hypothesis

Our experiments reveal a hierarchy of importance for GRPO hyperparameters on mathematical reasoning:

1. **Training data difficulty** (hard vs. all problems): +5.0pp on MATH-500

2. **Max completion length** (4096 vs. 2048 tokens): contributes to the same +5.0pp (confounded with data difficulty in our experiments)

3. **Group size** ($K = 16$ vs. 8): contributes to the 2.4–3.0pp gap between cloud and local LoRA

4. **LoRA vs. full-parameter**: <1pp difference when controlling for generation budget

5. **Continued training** (v3 vs. v2): +0.4pp with possible regression on some benchmarks

This ordering suggests that practitioners with limited compute should prioritize **longer generation context and larger group sizes** over more expensive parameterization (full-parameter training) or more training epochs.

7.3 Training Efficiency: LoRA's Advantage

LoRA's faster convergence in the RL setting (50 steps vs. 998 for full-parameter) deserves further analysis. We observe that:

- LoRA constrains updates to a low-rank subspace, providing implicit regularization that prevents the model from making large, destructive updates to its weights.
- The higher learning rate (4×10^{-5} vs. 5×10^{-6}) compensates for the smaller effective parameter count, enabling rapid adaptation.
- The low-rank constraint may focus learning on the most important directions in weight space, leading to more sample-efficient updates.

However, LoRA's fast convergence is a double-edged sword: training beyond the optimal checkpoint (step 50 in our setting) leads to overfitting, as evidenced by the MATH-500 degradation from 87.2% (step 50) to 86.0% (step 100). Practitioners using LoRA for RL should monitor gradient norms and reward saturation for early stopping.

7.4 Limitations

1. **Single model family.** All experiments use Qwen3-4B. Our findings about SFT degradation may not generalize to other architectures or model sizes.

2. **Small AIME sample size.** AIME benchmarks contain only $n = 30$ problems, yielding 95% CIs of ±14–18pp. Differences on AIME should be interpreted cautiously.

3. **Confounded ablations.** The data difficulty and context length changes between v1 and v2 were applied simultaneously. A fully factorial design would isolate these contributions more cleanly.

4. **Single-GPU constraints.** Our LoRA vs. full-parameter comparison is confounded by the different generation budgets necessitated by GPU memory. A multi-GPU experiment with matched hyperparameters would provide a cleaner comparison.

5. **Binary reward only.** We do not explore process reward models, partial credit, or reward shaping, which may unlock further gains.

8 Conclusion

We present a comprehensive study of post-training strategies for mathematical reasoning in a 4B-parameter language model. Our key findings are:

1. **RL over SFT:** GRPO with binary correctness reward improves MATH-500 accuracy by +10.8pp, while SFT degrades it by −2.2pp. SFT warm-starting provides no benefit for subsequent RL.

2. **Hard problems and long context:** Training exclusively on MATH Level 4–5 with 4096-token completions yields a +5.0pp improvement over training on all levels with 2048 tokens.

3. **LoRA efficiency:** Under single-GPU constraints, LoRA achieves comparable accuracy to full-parameter training in 50× fewer steps, with the generation budget (context length × group size) being more important than the parameterization choice.

4. **Inference-time scaling:** Majority voting at maj@16 boosts MATH-500 to 93.2% and AIME 2024 to 56.7%, with pass@64 revealing a 97.2% ceiling on MATH-500 that could be approached with a learned verifier.

These findings provide practical guidelines for post-training small language models: prioritize RL over SFT, train on hard problems with generous context budgets, use LoRA with early stopping, and leverage inference-time scaling to push beyond greedy decoding limits.

References

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., Liu, J., Qi, L., Liu, Z., and Sun, M. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS*, 2021.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

Lambert, N., Pyatkin, V., Morrison, J., Miranda, L., Lin, B. Y., Chandu, K., Dziri, N., Kumar, S., Zick, T., Choi, Y., Smith, N. A., and Hajishirzi, H. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *ICLR*, 2024.

Mukherjee, S., Mitra, A., Jawahar, G., Aber, S., Palangi, H., and Awadallah, A. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023.

Together AI. OpenR1-Math-220k. *Hugging Face dataset*, 2025.

QwQ Team. QwQ: Reflect deeply on the boundaries of the unknown. *Qwen Blog*, 2024.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Zhang, M., Li, Y., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Tinker. Tinker: Cloud fine-tuning API. https://tinker.dev, 2025.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., and Huang, S. TRL: Transformer reinforcement learning. https://github.com/huggingface/trl, 2022.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. MetaMath: Bootstrap your own mathematical questions for large language models. In *ICLR*, 2024.