



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING

DAPO: An Open Source LLM Reinforcement Learning System at Scale

Authors: Byte Dance Seed, Institute for AI Industry Research (Tsinghua University), SIA Lab (Tsinghua and ByteDance)

Outline

1. Background and Motivation
2. Introduction & Preliminaries
3. Method:
 - a. Clip Higher
 - b. **D**ynamic Sampling
 - c. Token Level **P**olicy Gradients
 - d. **O**verlong Reward Shaping
4. Experiments and Results
5. Connection to RL for Reasoning Problems

Open-Sourced [Dataset](#) and [Code](#) built on the **verl** framework for those interested.

Background and Motivation

The Problem

As we have already seen in the past week or so:

- RLHF has enabled us to get **longer CoT** and **stronger reasoning**.
- While the theory is known to us, RLHF is also a lot of:
 - **Algorithmic Tricks** and Empirical Choices / Dataset Specific
 - Too many **reward types** (Verifiable, Non-Verifiable Real-World Rewards)
 - Many ways to **stabilize training** (Clip or not to clip?, How to sample in GRPO?, KL or no KL?)

However, for us here in academia, this is mostly based on what we read. The **super smart** folks at OpenAI [1] and even DeepSeek [2] have mostly hidden methods that are essential for fixing issues like:

- **Entropy Collapse** (Mostly seen in GRPO)
- **Reward Noise**
- **Training Instability**

Background and Motivation

What did the authors do?

1. They started off by taking taking DeepSeek R1 paper (whatever they *did* report) and trying to replicate similar results by using **GRPO** on **Qwen2.5-32B**.
2. Results? **30 Points** on AIME. **DeepSeek claimed 47** on the same benchmark! Similar points noted by [3] and [4].

What do they propose?

1. Open-Source Large-Scale RLHF framework with techniques they found (some based in theory, some that I find mostly empirical) **to help with Long-CoT** reasoning in LLMs.
2. Claim that their method, **Decoupled Clip** and **Dynamic sAmpling Policy Optimization (DAPO)**, achieves **50 points** on AIME 2024 (+3) with **50% of training steps** used in DeepSeek-R1-Zero-Qwen-32B.

Introduction

They propose 4 key techniques:

CLIP HIGHER - Remember Clipping? They propose **increasing** the upper limit of the clip.

DYNAMIC SAMPLING - GRPO is only effective if each group *has some outputs with an **advantage***.
How to ensure that?

TOKEN-LEVEL Policy Gradient - Super key for **Long-CoT**. Longer responses should have more influence on overall gradient update

REWARD SHAPING - *Soft*
Punishment for longer responses.

And 2 minor one's:

PPO had the KL term. DPO made it implicit. GRPO had it implicitly, but still kept it. **They remove it.**

Super simple **verifiable reward** function. Just basically a binary reward function.

Preliminaries and Minor Changes

GRPO (Group Relative Policy Optimization)

How is it different from PPO? **No Value Function**. Advantage is estimated based on groups. **Advantage** of i-th response is:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}.$$

Similar to PPO in the sense that it still has the **clipped** objective and **KL Penalty**.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip} \left(r_{i,t}(\theta), 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right) - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \right],$$

where

$$r_{i,t}(\theta) = \frac{\pi_{\theta}(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}.$$

Sample Level Objective. **Mean Loss**
within generated sequence then
averaged for all samples.

Preliminaries and Minor Changes

DAPO (Decouple Clip and Dynamic Sampling Policy Optimization)

How is it different from GRPO? **No KL Term.**

Why? Long-CoT specifically **requires model to diverge** from initial model. Necessary to support longer reasoning.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip} \left(r_{i,t}(\theta), 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right) - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \right],$$

where

$$r_{i,t}(\theta) = \frac{\pi_{\theta}(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}.$$

They also change the reward model. Justification given is that it prevent **Reward Hacking**. Verifiable Reward based only **on the outcome** (There is also another reward component which comes later).

$$R(\hat{y}, y) = \begin{cases} 1, & \text{is_equivalent}(\hat{y}, y) \\ -1, & \text{otherwise} \end{cases}$$

Method: Clip Higher

Why is it needed?

PPO and GRPO suffer from **entropy collapse** - Entropy of policy decreases quickly with training. Sampled responses are identical! **In RL**, this is **exploration vs exploitation** case. Entropy Collapse causes low exploration and deterministic policy (Cause issues with scaling)!

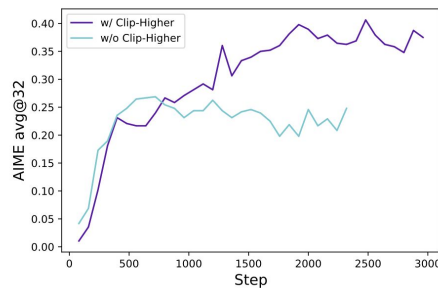
The authors propose that this is because the **upper clipping restricts exploration**.

Decoupled because E_high and E_low are different - fancy naming.

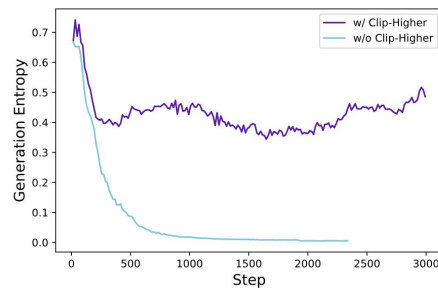
$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip} \left(r_{i,t}(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}} \right) \hat{A}_{i,t} \right) \right]$$

s.t. $0 < \left| \{o_i \mid \text{is_equivalent}(a, o_i)\} \right| < G.$

In practice, I just think this is an **empirical** choice. They take **E_High as 0.28** instead of the default 0.2 and keep E_Low as 0.2, the default.



(a) Accuracies on AIME.



(b) Entropy of actor model.

Method: Dynamic Sampling

Why is it needed?

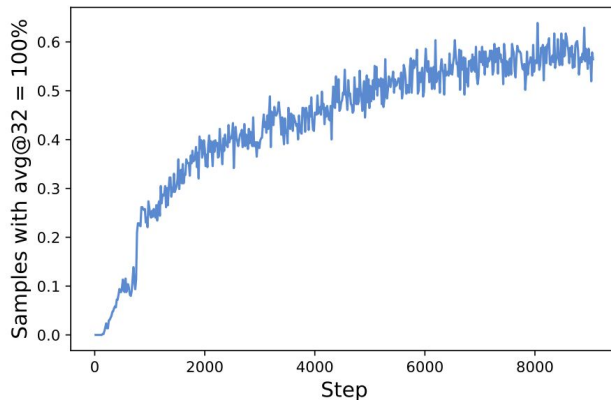
Gradient-Decreasing problem. Its normal for some prompt responses in the RLHF training process to have an accuracy of 1. But in GRPO, if all outputs of a group have accuracy 1, the **resulting advantage is 0**! Hence, **no gradients and reduced efficiency**.

The authors propose over-sample and then filter responses with accuracy as 1 or 0.

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip} \left(r_{i,t}(\theta), 1 - \varepsilon_{\text{low}}, 1 + \varepsilon_{\text{high}} \right) \hat{A}_{i,t} \right) \right]$$

s.t. $0 < \left| \{o_i \mid \text{is_equivalent}(a, o_i)\} \right| < G$.

I do think this is also because of the *overly simple reward function*. **Soft rewards** not just based on the outputs may help with this?



Method: Token-Level Policy Gradient Loss

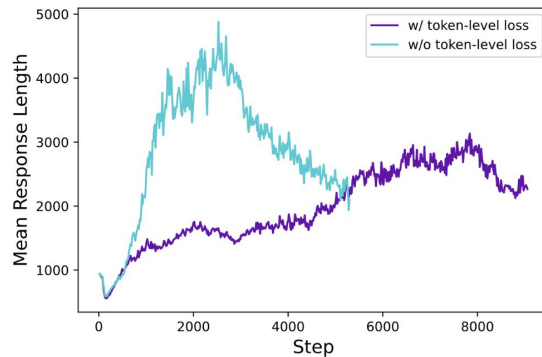
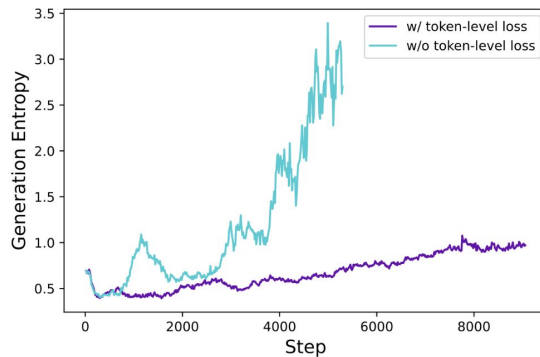
Why is it needed?

Original GRPO algorithm uses **sample-level loss** calculation. Average the losses by token within sample -> Aggregate losses across all samples. Issues?

- Each sample has equal contribution.
- If a response is overly long, it still has same loss contribution. **Good long responses** don't get any benefit and **bad long responses** don't get punished enough. Isn't optimized for Long-CoT!

$$\left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \right] \rightarrow \left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \right]$$

Gives **smoother entropy** and response length increase.



Method: Overlong Reward Shaping

Why is it needed?

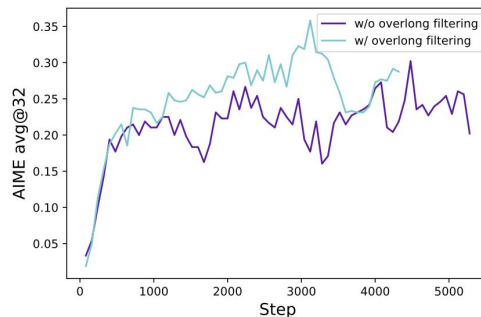
Typical RL training has fixed maximum length for generation - **Long samples truncated**. Authors say that is **counter-intuitive** for Long-CoT + **Introduces noise**.

Solution they tried was two fold:

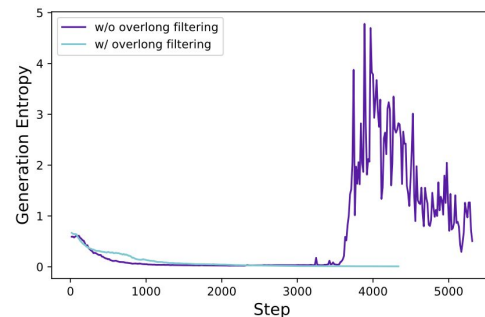
- **Overlong Filtering** - Mask the loss of truncated samples - no confusion in training process.
- **Soft overlong punishment** - Length-aware penalty for truncated samples. Longer the response, greater the punishment.

$$R_{\text{length}}(y) = \begin{cases} 0, & |y| \leq L_{\text{max}} - L_{\text{cache}} \\ \frac{(L_{\text{max}} - L_{\text{cache}}) - |y|}{L_{\text{cache}}}, & L_{\text{max}} - L_{\text{cache}} < |y| \leq L_{\text{max}} \\ -1, & L_{\text{max}} < |y| \end{cases}$$

Note the **interval**. In that interval, longer the response, greater the punishment. It is **dependent on actual answer's length**.



(a) Performance on AIME.



(b) Entropy of actor model.

Experiments and Results

Experimental Setup

Dataset - Sourced from Art of Problem Solving Website - Manual Annotation. Answers transformed to integers (Makes reward modeling simpler and matches AIME format). Open-sourced **DAPO-MATH-17k** dataset.

Training Setup

- AdamW with LR of $1e-6$. Linear warm-up over 20 rollouts.
- Prompt batch size is set to 512 and 16 responses sampled for each prompt.
- Max tokens is set to **16,384 (Expected)** and **L_cache of 4096** for soft punishment.
- **Clip High of 0.28** and Clip Low of 0.2. Results are **avg@32**.
- Temperature 1.0 and top-p of 0.7.

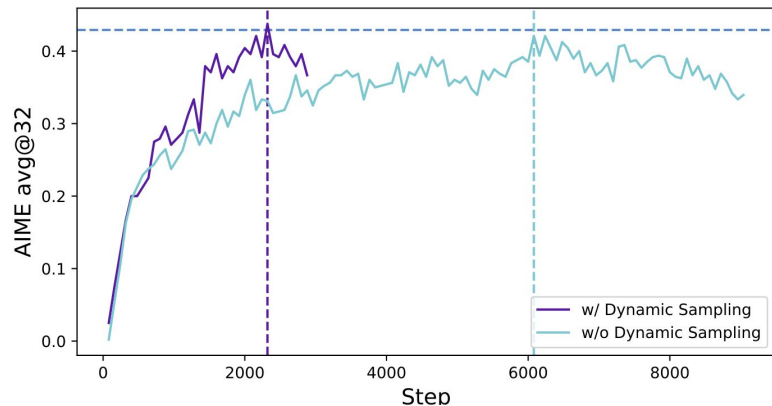
Dynamic Sampling technically needs more samples than naive GRPO -> However, in general you need fewer steps to converge so it's fine.

Note how token-level loss doesn't actually do much -> Authors say it **stabilizes training** though (?)

Table 1 Main results of progressive techniques applied to **DAPO**

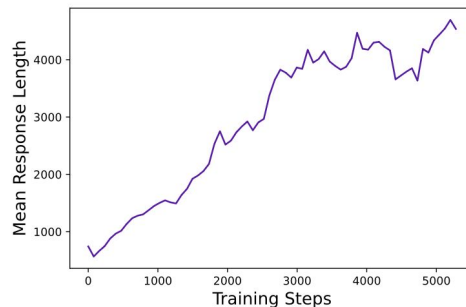
Model	AIME24 _{avg@32}
DeepSeek-R1-Zero-Qwen-32B	47
Naive GRPO	30
+ Overlong Filtering	36
+ Clip-Higher	38
+ Soft Overlong Punishment	41
+ Token-level Loss	42
+ Dynamic Sampling (DAPO)	50

Conclusions

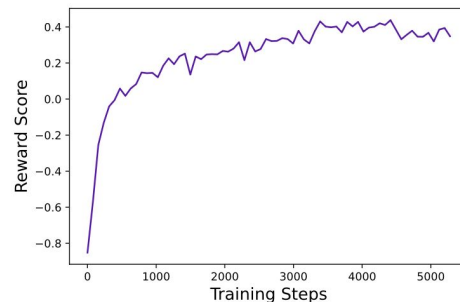


Surprisingly faster convergence. Would have been nice to see this plot for each of the ablations.

Elicits Long-Reasoning with more training steps. Note that correlation with accuracy shows this isn't gibberish just for the sake of being long.



Very stable reward increase. No fluctuation shows stable training.



Questions?

(and the next paper)



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING

Understanding R1-Zero-Like Training: A Critical Perspective

Authors: Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, Min Lin

* When (and why) RL is effective for reasoning problem?

Outline

1. Background & Motivation
2. Introduction & Goals
3. Analysis on Base Models
 - a. Role of Templates for Base Models to Function as a Question-Answering Base Policy
 - b. Qwen-2.5 Models Unlock the Best Performance When Discarding Template
 - c. Self-Reflection Already Appears in Base Models Including DeepSeek-V3-Base
4. Analysis on Reinforcement Learning
 - a. GRPO Leads to Biased Optimization
 - b. Dr. GRPO: Group Relative Policy Optimization Done Right
 - c. How Different Templates Affect the RL Training
 - d. Domain-Specific Pretraining Improves RL Ceiling
5. Conclusions

Background & Motivation

Language Model	Release	Base	Alignment Algorithm(s) Used
GPT-3-instruct	2020	GPT-3	SFT --> RLHF/PPO
GPT-4	2023	GPT-4 pre-trained?	SFT --> RLHF/PPO
Gemini	2023	Gemini pre-trained?	SFT --> RLHF/PPO
LLaMA2	2023	LLaMA2 pre-trained	SFT --> RLHF/PPO
LLaMA3	2024	LLaMA3 pre-trained	Iterate: Rejection sampling -> SFT -> DPO
Alpaca	2023	LLAMA 1	SFT
Qwen2.5	2024	Qwen2.5 pre-trained	SFT -> DPO -> GRPO
Tulu 3	2024	Llama 3.1	SFT -> DPO -> RLVR
DeepSeek (V3)	2024	DeepSeek pre-trained	SFT -> GRPO

What's common?

Background & Motivation

How DeepSeek-R1-Zero [1] differs and what they claim:

- **Drop SFT**, directly apply RL to base LLMs with simple rule-based or reward rules
 - No need for massive human-labeled data (QnA pairs, CoT examples etc.)
 - Avoid domain bottleneck and single-path bias
 - Use synthetic rules as reward functions (e.g., correctness checker for math, compiler for code)
- The whole pipeline reduces to: **base model** → **RL with cheap reward signals**
- Allows the model to explore chain-of-thought (CoT) for solving complex problems – How?
- No imitation from dataset; sample many candidate answers → verify using reward functions and REINFORCE!
- **Reward models/general rules generalize better than fixed labels**
- “Emergent behaviors” like self-reflection (“aha moment”), self-verification, longer CoT, etc., emerge during RL

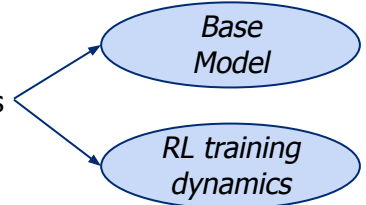
Background & Motivation

BUT, there were many open questions:

- How much of the reasoning was **already present** in the base model? *Maybe the model was already good, and RL just polished it or revealed preexisting capacities?*
- What exactly in the **RL algorithm** contributes to improvements? *Reward rules, the template over prompts, structure of question sets?*
- Are there optimization **artifacts or biases** that make things look better than they are? *(E.g. being rewarded for longer answers rather than correctness, etc.)*
- How much compute / how much “RL effort” is actually required; *can **simpler or “minimalist” versions** reach comparable performance?*

Introduction & Goals

At high level, the paper has three goals:

1. Dissect the R1-Zero paradigm into its two main components

```
graph LR; A[Dissect the R1-Zero paradigm into its two main components] --> B(Base Model); A --> C(RL training dynamics);
```

 - a. For free from the **pre-training**
 - b. Actually from the RL procedure
2. For each component, examine how much of the “reasoning ability” that R1-Zero claims comes

By asking these two questions:

- a. Do base models already show emergent reasoning / “aha moments,” self-reflection, etc., without RL?
 - b. Do certain model families (e.g. Qwen2.5, DeepSeek-V3-Base) have pretraining biases (e.g. training data, template usage) that make them more “ready” for R1-Zero style training?
3. Identify “hidden” biases or artifacts in the RL method used in R1-Zero (specifically GRPO). They may:
 - a. Inflate performance or
 - b. Inflate reasoning metrics in misleading ways

Analysis on Base Models - Take 1

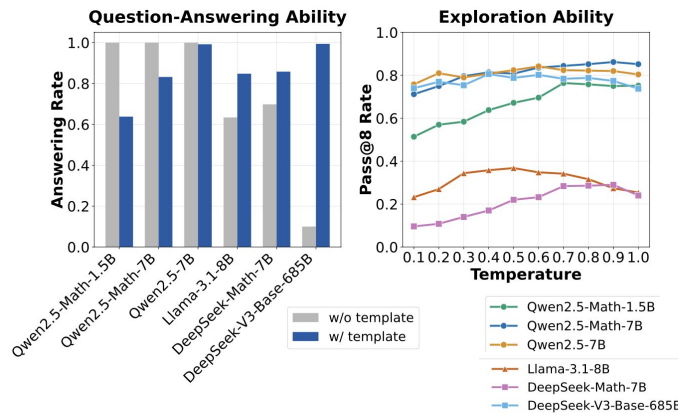
- Open-source base models → typically trained for sentence completion ($p_\theta(x)$), not Q&A
- To make them usable for R1-Zero, templates act like a “bridge”: they turn the model into a **question-answering policy** $\pi_\theta(\cdot | q)$
- Core Question:** Can the base models function as a question-answering base policy, given appropriate templates?

Template 1 (R1 template). A conversation between User and Assistant. The User asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the User with the answer. The reasoning process is enclosed within `<think>` `</think>` and answer is enclosed within `<answer>` `</answer>` tags, respectively, i.e., `<think>` reasoning process here `</think>` `<answer>` answer here `</answer>`.
User: {question}
Assistant: `<think>`

Template 2 (Qwen-Math template). `<|im_start|>system\nPlease reason step by step, and put your final answer within \\boxed{ }.<|im_end|>\n<|im_start|>user\n{question}<|im_end|>\n<|im_start|>assistant\n`

Template 3 (No template). {question}

- 500 questions sampled from the MATH [2] training set
- Asked **GPT-4o-mini** to judge whether the model responses are in an answering format



Analysis on Base Models - Take 2

- **Observation:** Qwen2.5 models work best when no template is used → warrants further investigation
- **Goal:** Evaluate the reasoning ability on five standard benchmarks

Base model + Template	AIME24	AMC	MATH500	Minerva	OlympiadBench	Avg.
Qwen2.5-Math-1.5B (4-shot prompting)	0.0	20.0	50.4	12.1	15.9	19.7
R1 template	0.0	9.6	21.2	6.6	2.2	7.9
Qwen template	20.0	32.5	33.0	12.5	22.8	24.2
No template	16.7	43.4	61.8	15.1	28.4	33.1
Qwen2.5-Math-7B (4-shot prompting)	3.3	22.5	61.6	10.7	20.9	23.8
R1 template	0.0	0.0	0.0	0.0	0.1	0.0
Qwen template	16.7	38.6	50.6	9.9	16.6	26.5
No template	0.2	45.8	69.0	21.3	34.7	38.2

Table 1: Qwen2.5-Math models might be pretrained on concatenated question-answer text, resulting in peak performance when **no template** is applied.

- Not using any template → improvement of about 60% compared to 4-shot prompting
- **Hypothesis based on the result:**
 - Qwen2.5-Math had chat-style question-answer pairs in its pretraining data
 - They might have pretrained on the concatenated text to maximize $\log p_{\theta}(q; o)$ directly
 - **The base models are already SFT-like without templates**

Analysis on Base Models - Take 3

- **Observation:** Lowest answering rate with no template suggests that DeepSeek-V3-Base is a nearly pure base model
- **Question:** Does a pure base model like DeepSeek-V3-Base demonstrate self-reflection?

Question(a): Find the least positive integer n such that $1/\sin 45^\circ \sin 46^\circ + 1/\sin 47^\circ \sin 48^\circ + \dots + 1/\sin 133^\circ \sin 134^\circ = 1/\sin n^\circ$.

Response

First, I notice that the left-hand side of the equation has the form $\sum_{k=1}^n \frac{1}{\sin a_k \sin a_{k+1}}$, where $a_1 = 45^\circ, a_2 = 47^\circ, \dots, a_{90} = 133^\circ$. I also notice that $\sin a_{k+1} = \sin(a_k + 2^\circ)$.

...

Simplifying, I get $\sin(3a) = 3 \sin a - 4 \sin^3 a$.

Aha! I can use this to get $\sin(a + 2b) = \sin((a + b) + b) = 3 \sin(a + b) \dots$

This looks messy, but I notice that it has the form $\sin a \cos b + \cos a \sin b$ again.

This gives me an idea. Maybe I can use this identity repeatedly to simplify the left-hand side of the equation.

...

Question(b): An element is randomly chosen from among the first 15 rows ...

Response

In Pascal's Triangle, every row starts and ends with 1, ...

...

This can be calculated as: *awkward silence* **Wait, I'm overthinking. Let's try again.** The number of elements in the first n rows of Pascal's Triangle is $(n + 1)! / [2! * (n - 1)!]$.

...

Question(c): Denali and Nate work for a dog walking business and are paid ...

Response

To solve this problem, let's start by defining the variables...

...

Again, since $x \neq 0$, this equation doesn't give us a valid solution either. **Let's check if we made an error. We should verify the problem conditions again.** The correct approach is to ...

...

Important follow-up: Does self-reflection actually make answers more accurate?

Experiments:

- Pick questions where the model sometimes reflects on itself
- For each such question, sample 100 answers
- Split them into two groups: w/ and w/o self-reflection
- Measure accuracy separately for both groups
- Compare accuracy difference between the two

Finding: In about half the cases, the self-reflective answers were not more accurate than the non-reflective ones.

Figure 4: Cases showing that DeepSeek-V3-Base already exhibits “Aha moment” even before RL tuning.

Analysis on RL - Length Bias in GRPO

$$\mathcal{J}_{GRPO}(\pi_{\theta}) = \mathbb{E}_{\mathbf{q} \sim p_{\mathcal{Q}}, \{\mathbf{o}_i\}_{i=1}^G \sim \pi_{\theta_{old}}(\cdot|\mathbf{q})}$$

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|\mathbf{o}_i|} \sum_{t=1}^{|\mathbf{o}_i|} \left\{ \min \left[\frac{\pi_{\theta}(\mathbf{o}_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(\mathbf{o}_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_{\theta}(\mathbf{o}_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(\mathbf{o}_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] \right\},$$

where

$$\hat{A}_{i,t} = \frac{R(\mathbf{q}, \mathbf{o}_i) - \text{mean}(\{R(\mathbf{q}, \mathbf{o}_1), \dots, R(\mathbf{q}, \mathbf{o}_G)\})}{\text{std}(\{R(\mathbf{q}, \mathbf{o}_1), \dots, R(\mathbf{q}, \mathbf{o}_G)\})},$$

- **Response-level length bias:** This arises from dividing by $|\mathbf{o}_i|$. How?
 - Case 1: Positive advantage ($A_{i,j} > 0$) -
 - Short correct answer contributes more per token to the gradient than longer correct answers
 - Result: model “learns” that if it wants to be rewarded strongly for being correct, it should output short answers
 - Case 2: Negative advantage ($A_{i,j} < 0$) -
 - Longer wrong answers gets less penalty per token than shorter wrong answers
 - Result: the model “learns” that if it’s going to be wrong, it’s safer (less penalized) to output longer responses

Analysis on RL - Difficulty Bias in GRPO

$$\mathcal{J}_{GRPO}(\pi_\theta) = \mathbb{E}_{\mathbf{q} \sim p_Q, \{\mathbf{o}_i\}_{i=1}^G \sim \pi_{\theta_{old}}(\cdot|\mathbf{q})}$$

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|\mathbf{o}_i|} \sum_{t=1}^{|\mathbf{o}_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_\theta(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] \right\},$$

where

$$\hat{A}_{i,t} = \frac{R(\mathbf{q}, \mathbf{o}_i) - \text{mean}(\{R(\mathbf{q}, \mathbf{o}_1), \dots, R(\mathbf{q}, \mathbf{o}_G)\})}{\text{std}(\{R(\mathbf{q}, \mathbf{o}_1), \dots, R(\mathbf{q}, \mathbf{o}_G)\})},$$

- **Question-level difficulty bias:** This arises from dividing by the standard deviation. How?
 - Case 1: std is small (everyone got similar reward), the advantage becomes large, question gets upweighted
 - Case 2: std is large (mixed correct/wrong answers), the advantage shrinks, question gets downweighted
- **Consequences:**
 - Too easy questions (everyone correct, std ≈ 0) \rightarrow upweighted (Model wastes effort reinforcing trivial questions)
 - Too hard questions (everyone wrong, std ≈ 0) \rightarrow also upweighted (Model spends lots of gradient on questions where it has no signal (since all answers are wrong))
 - Medium-difficulty questions (some right, some wrong, std high) \rightarrow downweighted (**These are actually the most useful questions (good learning signal), but GRPO gives them less weight!**)
 - Model's learning dynamics are **biased towards very easy or very hard questions**

Dr. GRPO: GRPO Done Right

- Simply **drop** the normalization terms
- Experiments on Qwen2.5-1.5B and R1 template with the following reward function:

$$R(\mathbf{q}, \mathbf{o}) = \begin{cases} 1 & \text{if } \mathbf{o} \text{ contains the correct final answer to } \mathbf{q} \\ 0 & \text{otherwise} \end{cases}$$

- Both exhibit similar trends for reward
- GRPO → Tends to continually generate longer responses even when the reward improvement slows down

The length of incorrect responses is substantially **reduced** by Dr. GRPO compared to the baseline (Plot 4).

This suggests that that an unbiased optimizer also **mitigates overthinking**.

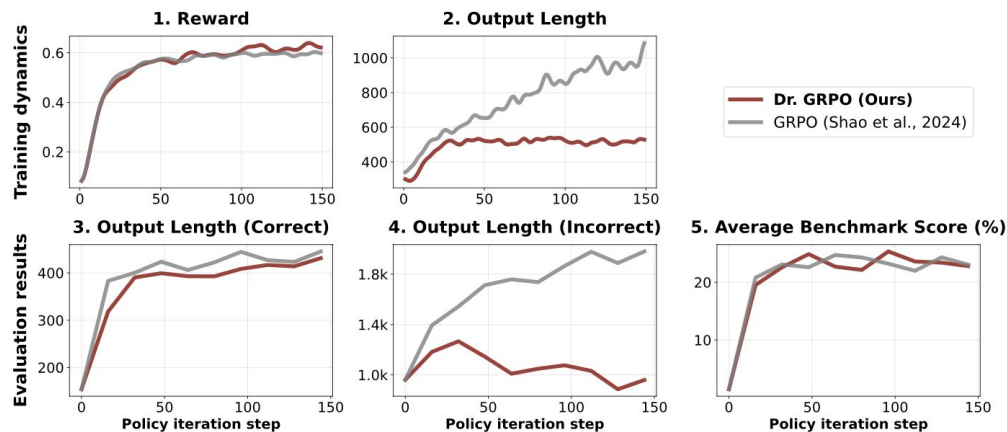


Figure 6: Comparison of Dr. GRPO and GRPO in terms of training dynamics (Top) and evaluation results (Bottom).

How Different Templates Affect RL Training

- **Motivation 1:** Qwen2.5-Math base models can answer questions with high accuracy without any prompt template
- **Motivation 2:** General belief: larger question set coverage leads to better performance - experiment with different templates and different levels of question coverage

1. RL can improve all policies to a comparable performance
2. R1 template - question sets have a significant impact on the dynamics of RL (lower coverage = lower performance)
3. Applying templates in fact destroys the capability before RL reconstructing it

Large mismatch between base models and templates → policy improvement mainly comes from **RL- tuning**

Else → even a small and completely o.o.d. question set could induce the reasoning ability equally well, by reinforcing correct reasoning behaviors instead of infusing new knowledge

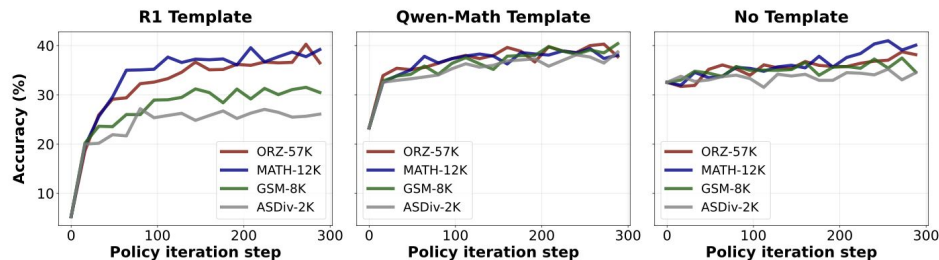


Figure 7: The average benchmark accuracy of different {template, question set} combinations during RL training.

Question set	Description
ORZ	Combining AIME, Numina-Math, Tulu3 MATH; diverse and large amount (57k)
MATH	High-school math competition questions (12k)
GSM	Simpler grade-school math questions (8k)
ASDiv	Basic algebra (+ − × ÷) questions (2k)

Table 3: Different question sets that have different levels of difficulty and coverage.

Can R1-Zero-like Training Succeed on Originally Weak Base-Models?

- **Motivation:** Recent R1-Zero-like replications mostly employ Qwen2.5 base models as the initial policies
- They are already strong math solvers and exhibit self-reflection patterns, as we've seen
- **What about originally weak base models?**
- **What about domain-specific pretraining?**
- Experiments on Llama-3.2-3B with Dr. GRPO and R1 template
- Llama-3.2-3B-FineMath - continual pretrained on the FineMath dataset
- To keep consistency with Qwen2.5 models, also prepare a concatenated dataset from NuminaMath-1.5 and continual pretrain it

1. RL works best when the base already has domain knowledge + Q&A knowledge
2. RL helps, but only a little on Vanilla LLaMA

GRPO → Both accuracy ↑ and length ↑ → misleading 'emergence' of longer CoT

Dr. GRPO → Accuracy ↑, length ↓
Accuracy ↓, length ↓ → **true improvement**

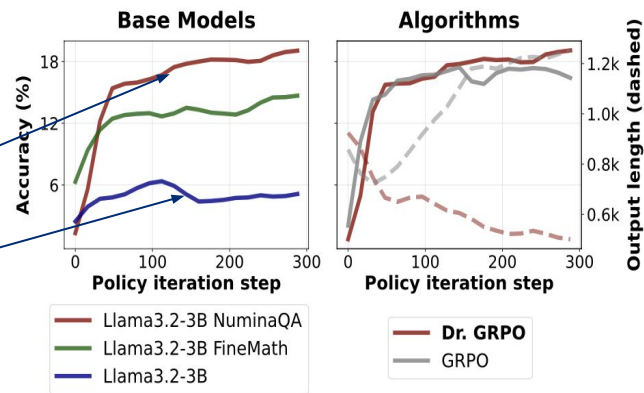






Figure 8: **Left:** The average benchmark performance curves of different base models. **Right:** The comparison between Dr. GRPO and GRPO with respect to reasoning accuracy (solid lines) and model response length (dashed lines).

Conclusions

So, when (and why) RL is effective for reasoning problem (this paper's context)?

-  RL is effective when the base model already has latent reasoning or domain knowledge
 - *E.g., Qwen2.5 pretraining → strong math skills + QA bias; LLaMA needs math pretraining to reach a usable ceiling*
-  RL reinforces and polishes existing reasoning behaviors, it does not create them from scratch
 - *Templates + base alignment matter: if mismatched, RL must work harder and needs broad coverage datasets*
-  RL must be optimized without bias
 - *GRPO can fake progress by rewarding length; Dr. GRPO corrects this, yielding true improvements*
-  Minimalist recipe works
 - *With the right base model + Dr. GRPO, even modest compute can reach SOTA on reasoning benchmarks*

Questions?



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING