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.

like DeepSeek-V3-Base demonstrates the Aha moment (Sec. 2.3). The middle plot of Fig. 3 shows the pass@8 accuracy of different base models (with template) at different sampling temperatures. This metric can serve as an indicator of base policy's exploration ability. For example, if a base policy cannot even sample a single trajectory that leads to the correct final answer, it is impossible for RL to improve the policy because there is no reward signal. Our results demonstrate that all tested models are exploratory (thus ready for RL), with Qwen2.5 models performing the best (even surpassing DeekSeek-V3-Base). This might partially explain that most R1-Zero projects (Zeng et al., 2025; Hu et al., 2025) are based on Qwen2.5 models.

2.2 Qwen-2.5 Models Unlock the Best Performance When Discarding Template

We next dig into the intriguing observation (c.f. Fig. 3(Left)) that all Qwen2.5 base models readily serve as chat models even without any template. We take a step further to evaluate the reasoning ability of Qwen2.5-Math models on five standard benchmarks: AIME 2024 (Li et al., 2024a), AMC (Li et al., 2024a), MATH500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). Following common practice, we use greedy decoding and limit the sampling budget to 3000 tokens.

As shown in Table 1, not using any template can drastically boost the average performance, resulting in an improvement of about 60% compared to the traditional 4-shot prompting. Since Qwen2.5-Math (Yang et al., 2024b) uses chat model's data (question-answer pairs) during the pretraining stage, we hypothesize that they might pretrain on the concatenated text to maximize $\log p_\theta(\mathbf{q}; \mathbf{o})$ directly. If our hypothesis turns out true, we shall be more careful about using Qwen2.5 models to reproduce DeepSeek-R1-Zero, since the base models are already SFT-like without templates.

2.3 Aha Moment Already Appears in Base Models Including DeepSeek-V3-Base

One of the most inspiring results of DeepSeek-R1-Zero is the emergence of self-reflection behaviors, a.k.a., Aha moment, through pure RL training. A few prior studies (Liu et al., 2025b; Yeo et al., 2025) have suggested that there may not be Aha moment in open-source R1 replications because the base models they use already exhibit self-reflection keywords. However, they have not tested DeepSeek-V3-Base, on which the real R1-Zero model was RL-tuned. We complete this missing piece by hosting DeepSeek-V3-Base-685B ourselves and investigating its responses to the 500 MATH questions with the R1 template. From the right plot of Fig. 3, we can observe that DeepSeek-V3-Base also generates a decent amount of self-reflections, further validating the claims of Liu et al. (2025b). We also show examples in Sec. E (Fig. 13) where DeepSeek-V3-Base generates keywords such as "Aha" and "wait".

An additional important question is whether self-reflection behaviors are associated with improved model performance after RL training. To investigate this, we host DeepSeek-R1-Zero and analyze its responses to the same questions from the MATH dataset. Although self-reflection behaviors occur more frequently in R1-Zero, we observe that these behaviors are not positively correlated with higher accuracy. Detailed analysis can be found in Sec. F.

3 Analysis on Reinforcement Learning

Language model generation can be formulated as a token-level Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p_{\mathcal{Q}})$. At each generation step t, the state $s_t \in \mathcal{S}$ is the concatenation of the input question and the output response generated so far: $s_t = \mathbf{q}; \mathbf{o}_{<t} = [q_1, \ldots, q_M, o_1, \ldots, o_{t-1}]$. The policy $\pi_\theta(\cdot|s_t)$ will select the next token o_t from the vocabulary $\mathcal{A}$, resulting in a deterministic transition to the next state $s_{t+1} = s_t; [o_t]$. The generation process starts from sampling an initial state $s_1 = \mathbf{q} \sim p_{\mathcal{Q}}$ from a set of questions, and stops when the autoregressive policy generates the `[eos]` token or exhausts the budget.

Typically, we maximize the entropy-regularized objective (Schulman et al., 2017a):

$$\mathcal{J}(\pi_\theta) = \mathop{\mathbb{E}}_{\mathbf{q}\sim p_{\mathcal{Q}}} \left[\mathop{\mathbb{E}}_{\mathbf{o}\sim\pi_\theta(\cdot|\mathbf{q})} [R(\mathbf{q}, \mathbf{o})] - \beta \mathbb{D}_{KL}[\pi_\theta(\cdot|\mathbf{q}))||\pi_{\text{ref}}(\cdot|\mathbf{q})] \right], \tag{1}$$

where $R(\mathbf{q}, \mathbf{o}) = \sum_{t=1}^{|\mathbf{o}|} r(s_t, o_t)$ is the return (Sutton & Barto, 2018) of the trajectory $\mathbf{q}; \mathbf{o}$, and π_{ref} is a reference policy. The KL regularization term is usually adopted ($\beta > 0$) for reinforcement learning from human feedback (Christiano et al., 2017), where r is a **reward model** learned from data collected by π_{ref}. In this case, regularization helps prevent π_θ from deviating too far from the distribution where the reward model is accurate (Jaques et al., 2019; Stiennon et al., 2020). However, RL-tuning reasoning models typically employs **rule-based verifiers** as r (Lambert et al., 2024), eliminating the concerns of distributional shift. This allows us to remove the KL term, which not only saves the memory and computation required by π_{ref} during training, but also potentially leads to better performance for R1-Zero-like training (Hu et al., 2025). We will assume $\beta = 0$ throughout this paper.

Policy optimization algorithms. To optimize π_θ with the above objective (Eq. (1) with $\beta = 0$), Proximal Policy Optimization (PPO) (Schulman et al., 2017b) maximizes the following surrogate objective:

$$\begin{aligned} \mathcal{J}_{PPO}(\pi_\theta) = \mathbb{E}_{\mathbf{q}\sim p_{\mathcal{Q}}, \mathbf{o}\sim\pi_{\theta_{\text{old}}}(\cdot|\mathbf{q})} \\ \sum_{t=1}^{|\mathbf{o}|} \left\{ \min \left[\frac{\pi_\theta(o_t|\mathbf{q}, \mathbf{o}_{<t})}{\pi_{\theta_{\text{old}}}(o_t|\mathbf{q}, \mathbf{o}_{<t})} \hat{A}_t, \text{clip}\left(\frac{\pi_\theta(o_t|\mathbf{q}, \mathbf{o}_{<t})}{\pi_{\theta_{\text{old}}}(o_t|\mathbf{q}, \mathbf{o}_{<t})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right] \right\}, \end{aligned} \tag{2}$$

where $\pi_{\theta_{\text{old}}}$ is the policy before the update, ϵ is the clipping hyperparameter, and $\hat{A}_t$ is an estimator of the advantage function of the t-th token. A standard way to estimate $\hat{A}_t$ is to compute the Generalized Advantage Estimation (GAE) (Schulman et al., 2015) with a learned value model V_ϕ. However, in the context of LLM RL-tuning, learning the value model is computationally expensive, so methods that estimate $\hat{A}_t$ without V_ϕ are practically preferred. For example, Shao et al. (2024) proposed GRPO, which first samples a group of responses $\{\mathbf{o}_1, \ldots, \mathbf{o}_G\}$ per question and computes their returns $\mathbf{R} = \{R_1, \ldots, R_G\}$, then sets the advantage of all tokens from $\mathbf{o}_i$ as $\hat{A}_t = \frac{R_i - \text{mean}(\mathbf{R})}{\text{std}(\mathbf{R})}$.

3.1 GRPO Leads to Biased Optimization

In Deepseek-R1-Zero (Guo et al., 2025), a notable trend is the consistent increase in response length throughout the training process. This is frequently interpreted as an indication of the development of advanced reasoning abilities such as self-reflection. Recent studies (Pan et al., 2025; Zeng et al., 2025; Hu et al., 2025) have replicated this phenomenon using various algorithms and implementations. However, we argue that **the observed increase in response length may also be attributed to a bias inherent in the GRPO (Shao et al., 2024) objective function**:

$$\begin{aligned} \mathcal{J}_{GRPO}(\pi_\theta) = \mathbb{E}_{\mathbf{q}\sim p_{\mathcal{Q}}, \{\mathbf{o}_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(\cdot|\mathbf{q})} \\ \frac{1}{G}\sum_{i=1}^{G} \frac{1}{|\mathbf{o}_i|} \sum_{t=1}^{|\mathbf{o}_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})} \hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}, 1-\epsilon, 1+\epsilon \right) \hat{A}_{i,t} \right] \right\}, \end{aligned} \tag{3}$$

where

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

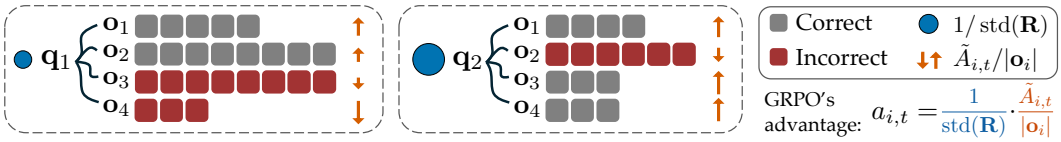

Figure 4: Illustration of the biases in GRPO. Note that the effective advantage of GRPO $a_{i,t}$ is equivalent to a reweighted version of the unbiased advantage $\tilde{A}_{i,t} = R(\mathbf{q}, \mathbf{o}_i) - \text{mean}(\mathbf{R})$. The terms $\text{std}(\mathbf{R})$ and $|\mathbf{o}_i|$ could bias the optimization by assigning different weights to different questions and responses, as denoted by the sizes of the blue circles and the lengths of the orange arrows. Upward arrows indicate positive advantages, and vice versa.

with the return $R(\mathbf{q}, \mathbf{o}_i)$ typically only including the *outcome verifiable reward* in LLM reasoning (the analysis also applies to process reward cases).

Compared to the objective function in Eq. (2), GRPO introduces two biases (see also Fig. 4):

- **Response-level length bias**: This arises from dividing by $|\mathbf{o}_i|$. For positive advantages ($\hat{A}_{i,t} > 0$, indicating a correct response), this bias results in greater gradient updates for shorter responses, leading the policy to favor brevity in correct answers. Conversely, for negative advantages ($\hat{A}_{i,t} < 0$, indicating an incorrect response), longer responses are penalized less due to their larger $|\mathbf{o}_i|$, causing the policy to prefer lengthier responses among incorrect ones.
- **Question-level difficulty bias**: This is caused by dividing the centered outcome reward by $\text{std}(\{R(\mathbf{q}, \mathbf{o}_1), \ldots, R(\mathbf{q}, \mathbf{o}_G)\})$. Questions with lower standard deviations (e.g., those that are too easy or too hard, with the outcome rewards being almost all 1 or 0) are given higher weights during policy updates. While advantage normalization is a common trick in RL (Andrychowicz et al., 2021), it is typically computed across an entire batch. In contrast, question-level normalization results in varying weights in the objective for different questions, leading to a difficulty bias in optimization.

Length Bias Also Exists in Open-Source PPO Implementations. We also examined several popular open-source implementations of vanilla PPO algorithms for LLM post-training. To our surprise, all of these implementations normalize the loss by response length (see Listing 1 and Table 2), which **misaligns** with the PPO objective as defined in Eq. (2). This formulation-implementation misalignment was present even before the publication of GRPO. We speculate that the misalignment might originate from the *pretraining stage* (Shoeybi et al., 2019), where all tokens are packed into a fixed-length context and normalizing the loss by the context length (i.e., computing `loss.mean(-1)`) improves the numerical stability. However, in the *RL-tuning stage*, typical implementations (von Werra et al., 2020) normalize the loss by the response length, which is **not** a constant, introducing an unintended length bias.

Listing 1: Comparison between a typical open-source PPO loss implementation that is biased (red) and our implementation (green). `MAX_TOKENS` is a global constant during the entire training (unless budget curriculum is enabled), which specifies the maximum number of generation tokens. Other constants also work with differences in gradient norm.

```
def masked_mean(tensor, mask, dim):
-    return (tensor * mask).sum(axis=dim) / mask.sum(axis=dim)
+    return (tensor * mask).sum(axis=-1) / MAX_TOKENS

ppo_loss = ...       # compute per-token ppo loss
response_mask = ...  # per-token response mask
# per-response length normalization (e.g., OpenRLHF)
loss_variant1 = masked_mean(ppo_loss, response_mask, dim=-1).mean()
# OR per-batch length normalization (e.g., trl, verl)
loss_variant2 = masked_mean(ppo_loss, response_mask, dim=None).mean()
```

3.2 Dr. GRPO: Group Relative Policy Optimization Done Right

To avoid the aforementioned optimization bias in GRPO, we propose to simply remove the $\frac{1}{|\mathbf{o}_i|}$ and $\text{std}(\{R(\mathbf{q}, \mathbf{o}_1), \ldots, R(\mathbf{q}, \mathbf{o}_G)\})$ normalization terms. Meanwhile, to faithfully