The Surprising Effectiveness of Negative Reinforcement in LLM Reasoning

Anonymous Authors[1]

Abstract

Reinforcement learning with verifiable rewards (RLVR) is a promising approach for training language models (LMs) on reasoning tasks that elicit emergent long chains of thought (CoTs). Unlike supervised learning, it updates the model using both correct and incorrect samples via policy gradients. To better understand its mechanism, we decompose the learning signal into reinforcing correct responses and penalizing incorrect ones, referred to as **P**ositive and **N**egative **S**ample **R**einforcement (**PSR** and **NSR**), respectively. We train `Qwen2.5-Math-7B` and `Qwen3-4B` on a mathematical reasoning dataset and uncover a surprising result: training with only negative samples—without reinforcing correct responses—can be highly effective: it consistently improves performance over the base model across the entire Pass@k spectrum (k up to 256), often matching or surpassing PPO and GRPO. In contrast, reinforcing only correct responses improves Pass@1 but degrades performance at higher k, due to reduced diversity. These inference-scaling trends highlight that solely penalizing incorrect responses may contribute more to performance than previously recognized. Through gradient analysis, we show that NSR works by suppressing incorrect generations and redistributing probability mass toward other plausible candidates, guided by the model's prior beliefs. It refines the model's existing knowledge rather than introducing entirely new behaviors. Building on this insight, we propose a simple variant of the RL objective that upweights NSR, and show that it consistently improves overall Pass@k performance on MATH, AIME 2025, and AMC23.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Language models (LMs) have recently demonstrated remarkable capabilities in various complex reasoning tasks, including mathematics (Cobbe et al., 2021; Hendrycks et al., 2021), coding (Jimenez et al., 2024; Yang et al., 2024b), and scientific reasoning (Phan et al., 2025; Rein et al., 2024). A key technique in achieving such success is reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025; Jaech et al., 2024; Lambert et al., 2024a; Team et al., 2025a), which is particularly effective in domains where the correctness of an outcome can be automatically verified via tools or functions. RLVR typically employs a binary reward ($+1$ or -1) based on the objective correctness of model responses. This simple yet effective mechanism not only mitigates reward hacking (Miao et al., 2024; Skalse et al., 2022) but also eliminates the need for extensive human annotations and complex reward model training (Li & Li, 2025; Zhang et al., 2025b).

RLVR's appeal is multifaceted: it offers a conceptually simple formulation (Lambert et al., 2024a), exhibits notable sample efficiency (Fatemi et al., 2025; Li et al., 2025; Wang et al., 2025), and enables inference-time scaling behaviors (Gandhi et al., 2025; Muennighoff et al., 2025; Wu et al., 2025b; Yeo et al., 2025; Zeng et al., 2025a). However, the precise mechanisms driving its effectiveness remain underexplored, particularly how it utilizes correct and incorrect samples. To address this, we decompose RLVR into two learning paradigms: *Positive Sample Reinforcement (PSR)* and *Negative Sample Reinforcement (NSR)*, as illustrated in Figure 1. This decomposition prompts a natural question: What roles do PSR and NSR play in shaping model behavior and generalization?

To empirically isolate the effects of PSR and NSR, we train two LMs, `Qwen2.5-Math-7B` and `Qwen3-4B`, either PSR or NSR exclusively, and evaluate their inference-time performance across a range of Pass@k metrics. We find that PSR-only training improves Pass@1 but hurts Pass@k at larger k values, indicating a loss of output diversity and exploration capacity. More strikingly, NSR-only training consistently improves performance over the base LM across the entire Pass@k spectrum and, in many cases, matches or even surpasses the performance of PPO (Schulman et al., 2017) and GRPO (Guo et al., 2025; Shao et al., 2024).

To understand why NSR alone is so effective, we conduct a token-level gradient analysis and demonstrate that NSR

works by suppressing incorrect reasoning steps and redistributing probability mass towards other plausible candidates already favored by the model's prior. This effectively refines its existing knowledge without aggressively teaching new behaviors. PSR, in contrast, sharpens the output distribution around sampled correct paths (Anthony et al., 2017; Havrilla et al., 2024; Xiong et al., 2025), often at the cost of suppressing alternative valid solutions. This suggests that NSR plays a pivotal role in preserving diversity and promoting generalization, especially when the base model already encodes strong reasoning priors.

Motivated by this insight, we propose Weighted-REINFORCE, a simple yet effective variant of the REINFORCE (Williams, 1992) objective that upweights its NSR contribution. We demonstrate that this adjustment consistently improves the Pass@k performance on MATH, AIME 2025, and AMC23, yielding an overall better result than strong baselines including PPO (Schulman et al., 2017) and GRPO (Guo et al., 2025; Shao et al., 2024).

Our contributions in this work are as follows:

- We decompose RLVR into two components, PSR and NSR, and investigate their distinct impacts on model behavior and generalization measured by a range of Pass@k metrics.
- We empirically demonstrate the surprising effectiveness of NSR-only training and use gradient analysis to show that NSR refines the model's prior by suppressing incorrect reasoning steps and preserving plausible alternatives.
- We propose Weighted-REINFORCE, a simple modification to the RL objective that upweights NSR, yielding consistent gains across complex reasoning benchmarks including MATH, AIME 2025, and AMC23.

2. RLVR Objective and Decomposition

2.1. Reinforcement Learning with Verifiable Rewards

Reinforcement learning with verifiable rewards (RLVR) has recently emerged as a powerful paradigm for enabling LMs to self-improve on tasks with objectively verifiable outcomes. The reward is given by a deterministic verification function r which assesses whether the model's response $\boldsymbol{y}$ to the prompt $\boldsymbol{x}$ is correct or not. All tokens $\{y_1, y_2, \ldots, y_T\}$ in a response $\boldsymbol{y}$ receive the same reward (*e.g.*, $+1$ for correct responses and -1 for incorrect ones).

Formally, given an LM with parameters θ, a set of prompts $\mathcal{D}$ from which $\boldsymbol{x}$ is sampled, and a verifiable reward function r, RLVR learns a policy π_θ to minimize the following objective:

$$\mathcal{L}_{\text{RLVR}}(\theta) = -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D},\boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}[r(\boldsymbol{x},\boldsymbol{y})], \quad r(\boldsymbol{x},\boldsymbol{y}) \in \{-1,+1\}. \tag{1}$$

In common RL algorithms (*e.g.*, PPO (Schulman et al., 2017), GRPO (Guo et al., 2025; Shao et al., 2024)), rewards are further normalized to stabilize training, enforcing a zero mean per batch $\mathcal{B}$ (*i.e.*, $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{B},\boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}[r(\boldsymbol{x},\boldsymbol{y})] = 0$).

2.2. Decomposing RLVR into Positive and Negative Sample Reinforcement

While RLVR has demonstrated promising empirical results, its underlying learning dynamics remain underexplored. In particular, it is unclear how the model updates its behavior under this binary outcome reward setting. What the model learns when receiving both positive and negative rewards can be entangled and hard to interpret. To better understand these dynamics, we begin by decomposing the RLVR objective into two distinct learning paradigms: learning from correct responses and learning from incorrect responses. This decomposition allows us to investigate how positive and negative reward signals shape model behavior, which we will show in the next section.

The RLVR objective optimizes the expected reward-weighted likelihood:

$$\begin{aligned}\mathcal{L}_{\text{RLVR}}(\theta) &= -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\sum_{\boldsymbol{y}} r(\boldsymbol{x},\boldsymbol{y})\cdot\pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right] \\ &= \underbrace{-\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\sum_{\boldsymbol{y}:r(\boldsymbol{x},\boldsymbol{y})=1}\pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right]}_{\mathcal{L}_{\text{PSR}}(\theta)} \\ &\quad \underbrace{-\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\sum_{\boldsymbol{y}:r(\boldsymbol{x},\boldsymbol{y})=-1}-\pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right]}_{\mathcal{L}_{\text{NSR}}(\theta)}, \\ & r(\boldsymbol{x},\boldsymbol{y}) \in \{-1,+1\},\end{aligned} \tag{2}$$

where we define two sub-objectives representing each learning paradigm:

$$\mathcal{L}_{\text{PSR}}(\theta) = -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\sum_{\boldsymbol{y}:r(\boldsymbol{x},\boldsymbol{y})=1}\pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right], \tag{3}$$

$$\mathcal{L}_{\text{NSR}}(\theta) = -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\sum_{\boldsymbol{y}:r(\boldsymbol{x},\boldsymbol{y})=-1}-\pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right]. \tag{4}$$

We refer to these two learning paradigms as *positive sample reinforcement* (PSR) and *negative sample reinforcement* (NSR). The positive reward case resembles supervised fine-tuning (SFT), where the model is updated to increase the likelihood of correct responses. In contrast, the negative reward case mirrors likelihood minimization that reduces the probability assigned to incorrect responses. Notably, PSR and NSR are on-policy—the responses are sampled

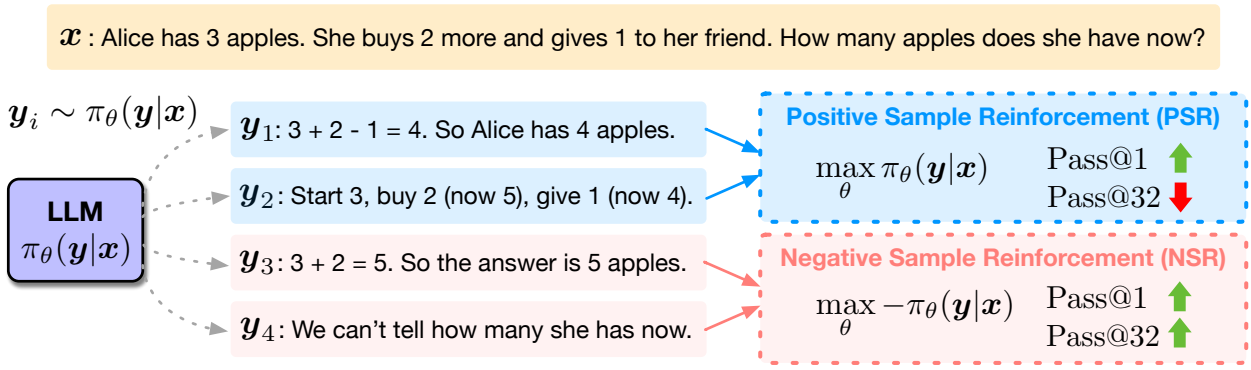

Figure 1: Decomposing learning signals in RLVR into positive and negative reward components. Positive Sample Reinforcement (PSR) increases the likelihood of correct responses and improves Pass@1, but reduces output diversity and hurts Pass@k for large k. Negative Sample Reinforcement (NSR) discourages wrong responses and redistributes probability mass according to the model's prior knowledge, improving the full Pass@k spectrum.

from the model itself during training.

Thus, the full RLVR objective decomposes into two sub-objectives $\mathcal{L}_{\text{RLVR}}(\theta) = \mathcal{L}_{\text{PSR}}(\theta) + \mathcal{L}_{\text{NSR}}(\theta)$. This decomposition reveals that RLVR jointly performs PSR on positively rewarded samples and NSR on negatively rewarded ones. To better understand the individual effects of these two learning paradigms on model behavior, we conduct experiments to train LLMs with each sub-objective independently, as well as the full RLVR objective for comparison.

3. Positive and Negative Sample Reinforcement for LLM Reasoning

3.1. Experimental Setup

Models. To understand how different training objectives affect the model's behavior, we train the model with PSR and NSR and evaluate their inference scaling performance on reasoning tasks. Specifically, we use `Qwen2.5-Math-7B` (Yang et al., 2024a) and `Qwen3-4B` (Yang et al., 2025) as the base models. `Qwen3-4B` has two modes (thinking and non-thinking), and we use the non-thinking mode for training and inference.

Compared algorithms. We compare the performance of PSR and NSR with commonly used RL algorithms, including PPO (Schulman et al., 2017) and GRPO (Guo et al., 2025; Shao et al., 2024). PSR and NSR are implemented by selectively updating the policy model using only correct or incorrect responses, respectively. As a result, PSR and NSR are trained on fewer samples per batch than standard RL algorithms (*e.g.*, PPO and GRPO) that use both correct and incorrect responses. The training objectives of these algorithms can be found in Appendix D.1. We also report the performance of the base models for reference.

Training setup. For the training set, we use MATH (Hendrycks et al., 2021), which contains 7,500 problems. We train the models using the verl framework (Sheng et al., 2024). The prompt batch size is 1,024, with 8 rollouts generated per prompt. The sampling temperature during training is set to 1.0, and the maximum context length is set to 4,096 and 32,768 tokens for `Qwen2.5-Math-7B` and `Qwen3-4B`, respectively. We update the model with a mini-batch size of 256 and a learning rate of 1e-6. More hyperparameter settings can be found in Appendix D.1.

Evaluation setup. We evaluate on three widely used math reasoning benchmarks, including the test sets of MATH, AIME 2025 and AMC23. During evaluation, we sample 256 responses per prompt for `Qwen2.5-Math-7B` with a temperature of 0.6 and a top-p of 0.95, and 64 responses for `Qwen3-4B` with a temperature of 0.7, a top-p of 0.8 and a top-k of 20, prompt templates can be found in Appendix D.2. For evaluation metric, recent work (Hochlehnert et al., 2025) highlights that accuracy based on greedy decoding can be unreliable. For this reason, we adopt a full spectrum of Pass@k as our main evaluation metric, using $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$ for `Qwen2.5-Math-7B` and $k \in \{1, 2, 4, 8, 16, 32, 64\}$ for `Qwen3-4B`. Pass@k is defined as the fraction of problems for which at least one correct response is produced in k independent trials. However, directly computing Pass@k using only k samples per example often suffers from high variance. We follow the unbiased estimator proposed by (Chen et al., 2021), which generates n samples per problem ($n \geq k$), counts the number of correct responses c, and computes an unbiased estimate of Pass@k as:

$$\text{Pass@}k = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right]. \tag{5}$$

Notably, varying k provides insights into different aspects of model behaviors. Pass@1 approximates greedy decoding accuracy, reflecting how confidently the model can produce a correct response in a single attempt, essentially reflecting *exploitation*. In contrast, Pass@k with large k evaluates

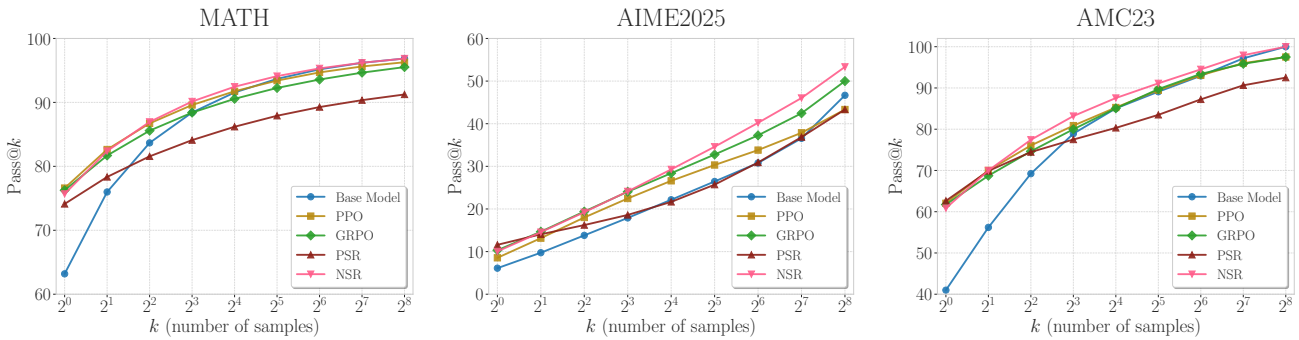

Figure 2: Pass@k curves of `Qwen2.5-Math-7B` trained with PPO, GRPO, PSR, and NSR. NSR is comparable to other methods across different k values and outperforms them at $k = 256$.

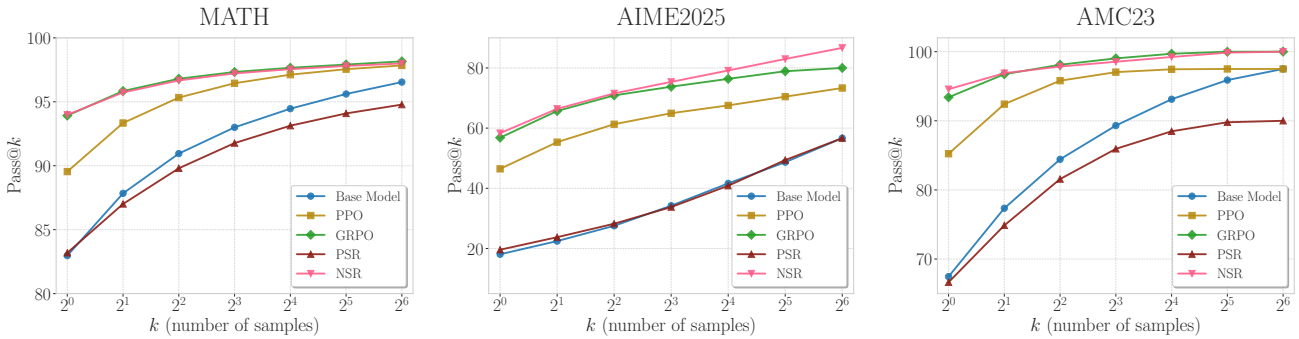

Figure 3: Pass@k curves of `Qwen3-4B` (non-thinking mode) trained with PPO, GRPO, PSR, and NSR. NSR consistently performs competitively across varying k values, while PSR does not improve the base model.

the model's ability to generate diverse correct responses across multiple attempts, capturing its *exploration* ability and reasoning boundary.

3.2. Inference Scaling Trends Under Different Training Objectives

NSR alone is surprisingly effective. As shown in Figures 2 and 3, NSR exhibits unexpectedly strong performance across the full range of k values. Despite being trained solely on negative samples, it consistently improves Pass@k compared to the base model. While PPO, GRPO, and PSR explicitly reinforce correct responses and naturally outperform the base model at Pass@1, it is surprising that NSR achieves a comparable Pass@1 without training on any correct responses. This suggests that NSR is able to reinforce correct responses indirectly by suppressing incorrect ones and redistributing probability mass toward plausible alternatives.

NSR outperforms or stays comparable to the base model at a large k value. At larger decoding budgets (*e.g.*, Pass@256), recent work (Yue et al., 2025) shows that RL-trained models often lose their advantage, and in some cases underperform the base model. This trend is generally observed in our experiments with PPO, GRPO, and PSR, especially in Figure 2. NSR, on the other hand, maintains a comparable or even better Pass@256 performance than the base model, generally outperforming the other algorithms. These results suggest that NSR promotes exploration and preserves the output diversity.

PSR improves accuracy at the cost of diversity. In contrast, PSR, which only reinforces correct samples, displays a more polarized behavior. It improves Pass@1, particularly on AIME 2025 and AMC23 in Figure 2. However, this precision comes at a cost: as k increases, Pass@k improves more slowly than other methods and eventually falls below the base model for $k > 8$. As shown in Figure 4a, PSR improves greedy decoding accuracy rapidly during early training but plateaus quickly and is overtaken by other methods. This behavior indicates that PSR overly concentrates probability mass on early correct responses, leading to overconfidence and a collapsed output distribution and ultimately limiting the model's ability to generate diverse correct responses when allowing for more test-time compute.

PSR fails to unlock the model's latent reasoning capabilities. Figure 3 demonstrates the performance of training `Qwen3-4B` under non-thinking mode to simulate a scenario where the model's underlying knowledge is known to be strong (*i.e.*, its learned thinking mode triggered by the '<think>' tag). While one may intuitively expect all algorithms to easily transfer the model's reasoning ability across

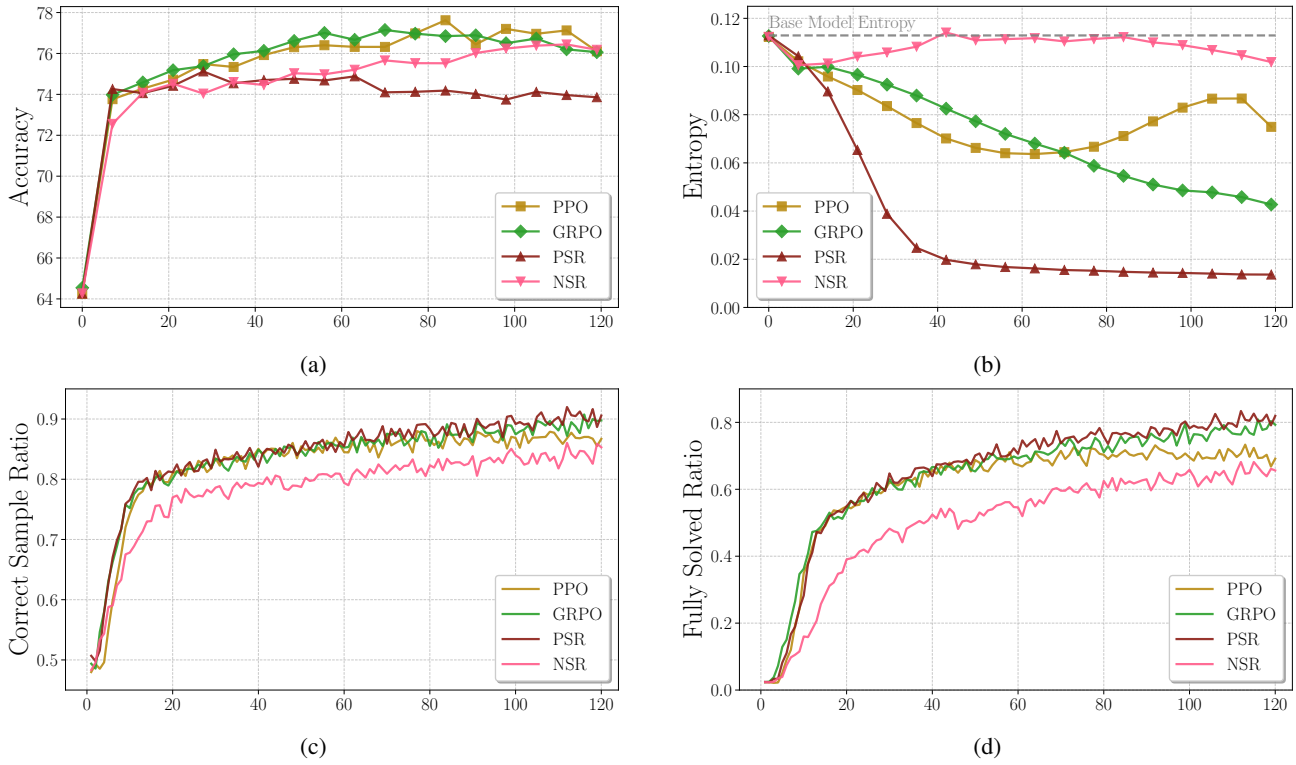

Figure 4: Training dynamics of `Qwen2.5-Math-7B` on MATH under PPO, GRPO, PSR and NSR across training steps, including (a) greedy decoding accuracy on the test set, (b) the model's entropy on the test set, (c) the ratio of correct responses per batch on the training set, and (d) the proportion of fully-solved prompts per batch (*i.e.*, all rollouts are correct) on the training set. NSR achieves competitive performance in greedy decoding accuracy while maintaining substantially higher entropy throughout training, suggesting greater exploration. NSR improves both the correct sample ratio and the fully-solved sample ratio, though less aggressively compared to other algorithms.

different prompt formats (*i.e.*, from thinking to non-thinking mode), our results demonstrate significant performance differences across different algorithms. PSR fails to activate these latent capabilities and does not improve the performance, even degrading Pass@k on MATH and AMC23 significantly. This highlights a fundamental limitation of PSR: it reinforces the currently dominant behavior in the output distribution, suppressing potentially stronger underlying capabilities. In contrast, NSR and GRPO significantly improve the performance of the base model across all Pass@k metrics, achieving thinking mode capabilities. Specifically, `Qwen3-4B` in thinking mode achieves a Pass@1 of 94.5 and a Pass@64 of 97.8 on MATH test set. NSR and GRPO closely match this performance, with NSR reaching 94.0 (Pass@1) and 98.0 (Pass@64), and GRPO achieving 93.9 (Pass@1) and 98.2 (Pass@64).

4. Understanding the Effectiveness of Negative Sample Reinforcement

To better understand the learning mechanisms of PSR and NSR, and to explain why NSR consistently demonstrates strong inference scaling performance, we analyze their training dynamics through both empirical observations and gradient analysis.

4.1. Entropy as a Lens on Inference Scaling Performance

Since diversity is crucial for strong Pass@k performance—especially at a large k—we quantify model diversity by tracking its entropy on a held-out test set throughout training, aiming to understand how entropy evolves under different training algorithms. In addition, we monitor two complementary metrics on the training set: the correct sample ratio, which measures the proportion of correct responses per batch, and the fully solved ratio, which measures the fraction of problems per batch where all rollouts are correct.

Figure 4b shows that NSR maintains a high level of entropy on the held out test set throughout training, closely matching that of the base model, while PSR leads to a rapid and substantial drop in entropy. PPO and GRPO offer a middle ground: they gradually reduce entropy during training, with levels that fall between those of PSR and NSR. Nevertheless, their entropy still declines considerably, which likely limits its output diversity and helps explain why their Pass@256

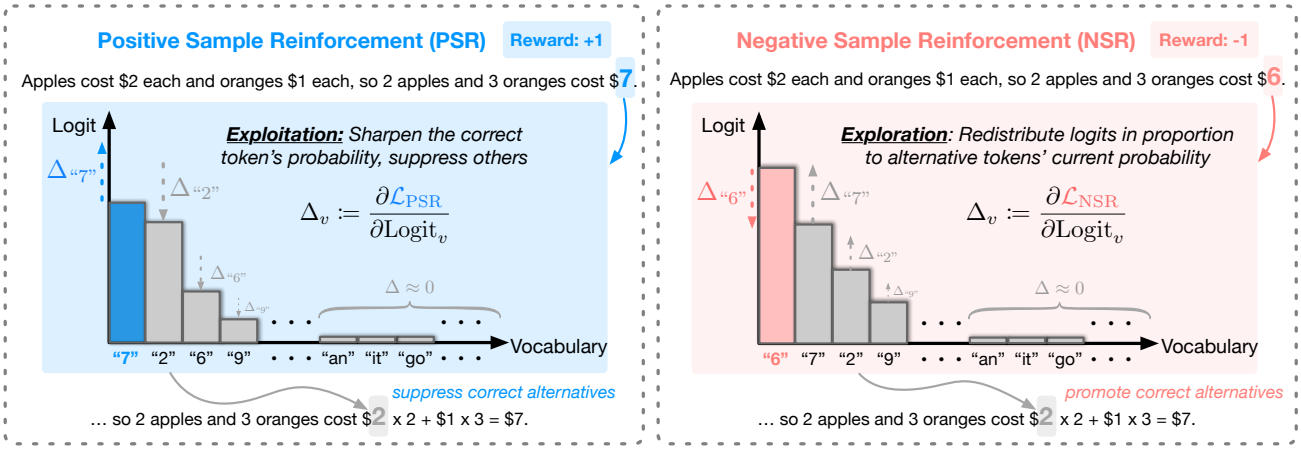

Figure 5: Gradient dynamics of PSR and NSR under a math word problem example. Bars indicate token logits before the update, and arrows indicate gradient direction and magnitude. Left (PSR): The model generates the correct response ("7") and receives a $+1$ reward. Gradients increase the logit of the sampled token while suppressing all others, including potentially correct alternatives like "2", resulting in a sharpened, overconfident distribution. Right (NSR): The model generates an incorrect response ("6") and receives a -1 reward. Gradients demote the sampled incorrect token and proportionally reallocate logits to other tokens (*e.g.*, "7", "2") based on their current probabilities, thereby promoting exploration on alternative correct paths and preserving diversity.

performance remains below that of the base model. Interestingly, PPO exhibits a slight rebound in entropy during the later stages of training—a trend not seen in GRPO. This divergence may stem from PPO's use of a critic model, which can provide potentially more fine-grained and exploratory advantage estimates.

As shown in Figures 4c and 4d, NSR consistently improves the correct sample ratio but maintains a lower proportion of correct samples and fully solved problems throughout training. This indicates that the model avoids becoming overconfident in the observed correct responses. By preserving uncertainty, NSR enables stronger scaling performance in Pass@k, as demonstrated in Figure 2. In contrast, PSR rapidly increases the number of correct samples in each batch and achieves a higher fully solved ratio compared to other methods, suggesting a tendency to overfit to the observed correct responses. While this leads to better Pass@1 performance, it significantly reduces output diversity and results in weaker Pass@k performance as k increases.

These trends underscore the importance of preserving output entropy during RL and highlight a key insight: reinforcing negative samples alone can improve accuracy without compromising the base model's generation diversity.

4.2. Token-Level Gradient Dynamics of PSR and NSR

To gain a deeper understanding of the training dynamics of PSR and NSR, we conduct a token-level gradient analysis. The loss for both PSR and NSR on a training instance $(\boldsymbol{x}, \boldsymbol{y})$ takes the form:

$$\begin{aligned}\mathcal{L}(\theta) &= -R \cdot \frac{1}{T}\sum_{t=1}^{T} \pi_\theta(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}) \\ &= -R \cdot \frac{1}{T}\sum_{t=1}^{T} \frac{\exp(z_{y_t})}{\sum_{v'\in\mathcal{V}}\exp(z_{v'})},\end{aligned} \tag{6}$$

where $R = r(\boldsymbol{x}, \boldsymbol{y}) \in \{-1, +1\}$ denotes the reward assigned to the sampled trajectory, and z_v denotes the logit corresponding to token v in the vocabulary $\mathcal{V}$.

To analyze the effect of these objectives on the model's token distribution, we compute the gradient of the loss with respect to token-level logits at each step. Let $\pi_v = \pi_\theta(v|\boldsymbol{x}, \boldsymbol{y}_{<t})$ denote the probability of token v at time step t, the gradient descent directions of PSR and NSR are shown in Equations (7) and (8) (the full derivation is provided in Appendix B), which are illustrated in Figure 5.

$$-\frac{\partial \mathcal{L}_{\text{PSR}}}{\partial z_v} \propto \begin{cases} \pi_{y_t} \cdot (1 - \pi_{y_t}) & \text{if } v = y_t \quad \text{(sampled token)} \\ -\pi_v \cdot \pi_{y_t} & \text{if } v \neq y_t \quad \text{(unsampled token)} \end{cases} \tag{7}$$

This formulation clearly shows how PSR increases the logits of tokens appearing in correct responses while decreasing the logits of all other tokens. As training progresses, it repeatedly amplifies the probability of observed correct sequences—pushing their likelihoods toward 1, while suppressing alternative generations. This continual sharpening of the output distribution can lead to reduced entropy and overfitting as shown in Figure 4b, especially in on-policy settings where the same examples may be encountered frequently. Over time, the model's behavior may collapse onto a narrow set of responses, limiting its ability to explore or

Key Insights into NSR via Gradient Analysis

1. **Preserving high-confidence priors**: When the model assigns high probability to certain tokens (i.e., $\pi_{y_t} \to 1$) that appear in incorrect outputs (*e.g.*, common grammatical or linguistic constructions), the negative gradient from NSR is scaled by $(1 - \pi_{y_t})$, resulting in small updates. This allows NSR to penalize mistakes without erasing fundamental knowledge learned during pretraining.
2. **Prior-guided probability redistribution**: NSR performs a soft reranking of the output distribution by boosting unsampled tokens' logits z_v in proportion to their current probabilities π_v. This allows the model to effectively explore and search for better candidates according to their prior beliefs.
3. **Implicit regularization against overfitting**: NSR updates only when the model generates incorrect responses. Once the model consistently avoids these mistakes, NSR naturally halts further updates on those samples. This stopping criterion prevents the model from overfitting or collapsing diversity in examples it has already mastered.

generalize beyond what has been reinforced.

$$-\frac{\partial \mathcal{L}_{\text{NSR}}}{\partial z_v} \propto \begin{cases} -\pi_{y_t} \cdot (1 - \pi_{y_t}) & \text{if } v = y_t \ \text{(sampled token)} \\ \pi_v \cdot \pi_{y_t} & \text{if } v \neq y_t \ \text{(unsampled token)} \end{cases} \tag{8}$$

In contrast, NSR works by penalizing the logits of tokens in incorrect responses and softly redistributing probability mass to other candidate tokens. Importantly, NSR increases other tokens' logits in proportion to their current likelihoods. This objective has several desirable properties:

These analyses highlight an important strength of NSR: it preserves the model's prior knowledge over plausible tokens and stops updating once errors are corrected, effectively locking in successful experiences. This is a unique advantage of NSR, as we demonstrate in Appendix B via gradient analysis that simply applying an entropy bonus in the policy loss, despite promoting diversity, cannot preserve model's prior.

4.3. Extending Gradient Analysis to PPO and GRPO

Our earlier analysis was based on a simple REINFORCE-like objective (Williams, 1992). We now analyze how it extends to PPO and GRPO. The losses for GRPO and PPO are as follows:

$$\mathcal{L}_{\text{PPO}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \min\left(\frac{\pi_\theta(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x}, \boldsymbol{y}_{<t})}, 1-\epsilon, 1+\epsilon\right) A_t\right), \tag{9}$$

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T}\sum_{t=1}^{T}\left(\min\left(\frac{\pi_\theta(y_{i,t}|\boldsymbol{x}, \boldsymbol{y}_{i,<t})}{\pi_{\text{old}}(y_{i,t}|\boldsymbol{x}, \boldsymbol{y}_{i,<t})} A_{i,t}, \text{clip}\left(\frac{\pi_\theta(y_{i,t}|\boldsymbol{x}, \boldsymbol{y}_{i,<t})}{\pi_{\text{old}}(y_{i,t}|\boldsymbol{x}, \boldsymbol{y}_{i,<t})}, 1-\epsilon, 1+\epsilon\right) A_{i,t}\right) - \beta \text{KL}(\pi_\theta \| \pi_{\text{ref}})\right), \tag{10}$$

where A denotes the advantage. Comparing the PPO and GRPO objectives to Equation (6), there are three key differences: (1) policy loss clipping, (2) KL regularization (reward penalty in PPO, loss in GRPO), and (3) advantages instead of raw rewards. We analyze their impact on the gradient dynamics below:

1. Clipping constrains update magnitude when the new policy diverges significantly from the old one, but preserves the gradient update direction, thus leaving our analysis qualitatively unchanged.
2. KL regularization discourages deviation from the reference policy. While it may dampen positive and negative updates, its practical impact is often negligible: for reasoning tasks, the KL coefficient is either very small or completely removed, leading to better performance, as demonstrated in recent works (Chu et al., 2025; Xu et al., 2025b; Yu et al., 2025).
3. The advantage of GRPO $A_i = \frac{r_i - \text{mean}(\boldsymbol{r})}{\text{std}(\boldsymbol{r})}$ acts as a rescaling of the gradient and retains the raw reward's sign. In PPO, the advantage is computed relative to a value function: tokens that outperform the baseline are reinforced, while other tokens are penalized. This yields finer-grained credit assignment but does not alter the overall gradient dynamics—positive reinforcement still reduces diversity, and negative reinforcement promotes alternatives under the model prior.

Therefore, while PPO and GRPO introduce modifications that stabilize learning, the core gradient behaviors identified in our previous analysis still hold. We further discuss the differences between RLVR and RL with reward models in Appendix C.

5. Balancing Positive and Negative Reinforcement

Our previous analyses reveal a trade-off in reinforcement learning objectives: PSR improves Pass@1 quickly but sacrifices performance on Pass@k for larger k, whereas

Table 1: Pass@k results on MATH, AIME 2025 and AMC23 with `Qwen2.5-Math-7B`. **Bold** and underlined numbers denote the best and second-best results for each k.

Method	**Pass@k**								
k	1	2	4	8	16	32	64	128	256
					MATH				
Base Model	63.2	76.0	83.7	88.4	91.6	93.7	95.2	**96.2**	**96.9**
GRPO	76.3	81.7	85.6	88.4	90.6	92.3	93.6	94.7	95.5
PPO	**76.6**	82.6	86.7	89.6	91.7	93.4	94.7	95.6	96.3
PSR	74.1	78.3	81.6	84.1	86.2	87.9	89.3	90.4	91.2
NSR	75.7	82.4	86.9	90.1	**92.4**	**94.1**	**95.3**	**96.2**	**96.9**
W-REINFORCE	**76.6**	**82.8**	**87.1**	**90.2**	**92.4**	**94.1**	**95.3**	96.1	96.7
					AIME 2025				
Base Model	6.1	9.7	13.8	17.9	22.2	26.5	30.8	36.6	46.7
GRPO	10.3	14.7	19.4	24.0	28.4	32.8	37.3	42.5	50.0
PPO	8.5	13.2	18.0	22.5	26.6	30.3	33.8	37.9	43.3
PSR	**11.6**	14.1	16.2	18.6	21.7	25.7	30.9	36.9	43.3
NSR	10.0	14.6	19.2	24.1	29.3	34.6	40.2	46.0	53.3
W-REINFORCE	10.6	**15.3**	**20.0**	**24.7**	**29.7**	**34.6**	**40.5**	**47.8**	**56.7**
					AMC23				
Base Model	41.0	56.2	69.2	78.9	85.1	89.1	92.9	97.2	**100.0**
GRPO	61.7	68.7	74.6	80.0	85.1	89.7	93.4	95.9	97.5
PPO	62.0	**70.0**	76.1	80.9	85.3	89.5	93.1	96.0	97.5
PSR	**62.6**	69.9	74.5	77.5	80.3	83.5	87.2	90.6	92.5
NSR	60.9	**70.0**	**77.4**	**83.2**	87.6	91.1	94.5	**97.9**	**100.0**
W-REINFORCE	62.0	**70.0**	77.0	83.1	**87.8**	**91.8**	**95.2**	97.1	97.5

NSR preserves Pass@k, but may underperform when k is small. To strike a better balance between accuracy and diversity, we propose a simple weighted combination of PSR and NSR: we scale down the reward magnitude for PSR by a factor of λ and combine it with NSR, allowing the model to learn from both correct and incorrect samples. When $\lambda = 1$, it is exactly the same as REINFORCE. We refer to this method as **Weighted-REINFORCE (W-REINFORCE)**:

$$\mathcal{L}_{\text{W-REINFORCE}}(\theta) = \underbrace{-\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\sum_{\boldsymbol{y}:r(\boldsymbol{x},\boldsymbol{y})=1} \lambda \cdot \pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right]}_{\lambda\cdot\mathcal{L}_{\text{PSR}}(\theta)} \underbrace{-\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\sum_{\boldsymbol{y}:r(\boldsymbol{x},\boldsymbol{y})=-1} -\pi_\theta(\boldsymbol{y}|\boldsymbol{x})\right]}_{\mathcal{L}_{\text{NSR}}(\theta)}. \tag{11}$$

We apply $\lambda = 0.1$ in our experiments. As shown in Table 1, W-REINFORCE consistently delivers a favorable trade-off across a range of k values on MATH, AIME 2025 and AMC23. On MATH, W-REINFORCE matches the best Pass@1 score 76.6 with PPO and has the highest performance for all $k \leq 64$, while still preserves competitive performance at $k = 256$. On AIME 2025, W-REINFORCE performs even more strongly—achieving the best result across all k values except $k = 1$. These results confirm that W-REINFORCE, a simple weighted extension of the classic REINFORCE algorithm, strikes an effective balance between the strengths of PSR and NSR. By merely scaling down the weight of positive rewards, it achieves strong performance while preserving diversity. Despite its simplicity, W-REINFORCE consistently outperforms strong RL algorithms such as PPO and GRPO across most Pass@k. These findings suggest this simple variant of REINFORCE can serve as a competitive alternative to more complex RL algorithms when the model prior is strong (*e.g.*, Qwen models).

6. Conclusion

In this work, we investigate the mechanism underlying RLVR for LM reasoning. By decomposing RLVR into positive and negative sample reinforcement, we reveal a surprising finding: solely penalizing incorrect samples can effectively enhance LM reasoning capabilities while preserving generation diversity. Experimental results show that NSR consistently improves performance across a wide Pass@k spectrum and in many cases matches or outperforms strong RL algorithms such as PPO and GRPO. Our gradient analysis demonstrates that NSR works by suppressing incorrect responses and redistributing probability mass toward plausible alternatives based on the model prior. Building on these findings, we proposed a simple variant of REINFORCE, Weighted-REINFORCE, that upweights the negative sample reinforcement. Empirical results show that it achieves a good balance between PSR and NSR, and yields consistent Pass@k improvements across multiple reasoning benchmarks. We discuss limitations and future work directions in Appendix E.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5366–5376, 2017.

Balachandran, V., Chen, J., Chen, L., Garg, S., Joshi, N., Lara, Y., Langford, J., Nushi, B., Vineet, V., Wu, Y., et al. Inference-time scaling for complex tasks: Where we stand and what lies ahead. *arXiv preprint arXiv:2504.00294*, 2025.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large Language Monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Chen, L., Davis, J. Q., Hanin, B., Bailis, P., Stoica, I., Zaharia, M. A., and Zou, J. Y. Are more LLM calls all you need? Towards the scaling properties of compound AI systems. *Advances in Neural Information Processing Systems*, 37:45767–45790, 2024.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating Large Language Models trained on code. *arXiv preprint arXiv:2107.03374*, abs/2107.03374, 2021.

Chu, X., Huang, H., Zhang, X., Wei, F., and Wang, Y. GPG: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*, 2025.

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

Dai, M., Yang, C., and Si, Q. S-GRPO: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025.

Dang, X., Baek, C., Wen, K., Kolter, Z., and Raghunathan, A. Weight ensembling improves reasoning in language models. *arXiv preprint arXiv:2504.10478*, 2025.

Dong, H., Xiong, W., Pang, B., Wang, H., Zhao, H., Zhou, Y., Jiang, N., Sahoo, D., Xiong, C., and Zhang, T. RLHF Workflow: From reward modeling to online RLHF. *arXiv preprint arXiv:2405.07863*, 2024.

Fatemi, M., Rafiee, B., Tang, M., and Talamadupula, K. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Gandhi, K., Chakravarthy, A., Singh, A., Lile, N., and Goodman, N. D. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

Gao, J., Xu, S., Ye, W., Liu, W., He, C., Fu, W., Mei, Z., Wang, G., and Wu, Y. On designing effective RL reward at training time for LLM reasoning. *arXiv preprint arXiv:2410.15115*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Havrilla, A., Du, Y., Raparthy, S. C., Nalmpantis, C., Dwivedi-Yu, J., Zhuravinskyi, M., Hambro, E., Sukhbaatar, S., and Raileanu, R. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 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 Datasets and Benchmarks*, 2021.

Hochlehnert, A., Bhatnagar, H., Udandarao, V., Albanie, S., Prabhu, A., and Bethge, M. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *arXiv preprint arXiv:2504.07086*, 2025.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. R. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

Kazemnejad, A., Aghajohari, M., Portelance, E., Sordoni, A., Reddy, S., Courville, A., and Roux, N. L. VinePPO: Unlocking RL potential for LLM reasoning through refined credit assignment. *arXiv preprint arXiv:2410.01679*, 2024.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. TÜLU 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024a.

Lambert, N., Pyatkin, V., Morrison, J., Miranda, L., Lin, B. Y., Chandu, K., Dziri, N., Kumar, S., Zick, T., Choi, Y., et al. RewardBench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024b.

Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K. R., Bishop, C., Hall, E., Carbune, V., Rastogi, A., et al. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Machine Learning*, pp. 26874–26901. PMLR, 2024.

Li, W. and Li, Y. Process reward model with Q-value rankings. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wQEdh2cgEk.

Li, X., Zou, H., and Liu, P. LIMR: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding R1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Meng, Y., Xia, M., and Chen, D. SimPO: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Miao, Y., Zhang, S., Ding, L., Bao, R., Zhang, L., and Tao, D. InfoRM: Mitigating reward hacking in RLHF via information-theoretic reward modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Mroueh, Y. Reinforcement learning with verifiable rewards: GRPO's effective loss, dynamics, and success amplification. *arXiv preprint arXiv:2503.06639*, 2025.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Phan, L., Gatti, A., Han, Z., Li, N., Hu, J., Zhang, H., Zhang, C. B. C., Shaaban, M., Ling, J., Shi, S., et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Qu, Y., Yang, M. Y., Setlur, A., Tunstall, L., Beeching, E. E., Salakhutdinov, R., and Kumar, A. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*, 2025.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct Preference Optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741, 2023.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Setlur, A., Rajaraman, N., Levine, S., and Kumar, A. Scaling test-time compute without verification or RL is suboptimal. *arXiv preprint arXiv:2502.12118*, 2025.

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

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.

Skalse, J., Howe, N., Krasheninnikov, D., and Krueger, D. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9460–9471, 2022.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Team, K., Du, A., Gao, B., Xing, B., Jiang, C., Chen, C., Li, C., Xiao, C., Du, C., Liao, C., et al. Kimi K1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025a.

Team, P. I., Jaghouar, S., Mattern, J., Ong, J. M., Straube, J., Basra, M., Pazdera, A., Thaman, K., Di Ferrante, M., Gabriel, F., et al. INTELLECT-2: A reasoning model trained through globally decentralized reinforcement learning. *arXiv preprint arXiv:2505.07291*, 2025b.

Wang, J., Fang, M., Wan, Z., Wen, M., Zhu, J., Liu, A., Gong, Z., Song, Y., Chen, L., Ni, L. M., et al. OpenR: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*, 2024a.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024b.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-Consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wang, Y., Yang, Q., Zeng, Z., Ren, L., Liu, L., Peng, B., Cheng, H., He, X., Wang, K., Gao, J., et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025.

Welleck, S., Bertsch, A., Finlayson, M., Schoelkopf, H., Xie, A., Neubig, G., Kulikov, I., and Harchaoui, Z. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.

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

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Wu, Y., Wang, Y., Du, T., Jegelka, S., and Wang, Y. When more is less: Understanding chain-of-thought length in LLMs. *arXiv preprint arXiv:2502.07266*, 2025b.

Xiong, W., Yao, J., Xu, Y., Pang, B., Wang, L., Sahoo, D., Li, J., Jiang, N., Zhang, T., Xiong, C., et al. A minimalist approach to LLM reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025.

Xu, Y., Dong, H., Wang, L., Sahoo, D., Li, J., and Xiong, C. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*, 2025a.

Xu, Y. E., Savani, Y., Fang, F., and Kolter, Z. Not all rollouts are useful: Down-sampling rollouts in LLM reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025b.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-Math technical report: Toward Mathematical Expert Model via Self-Improvement. *arXiv preprint arXiv:2409.12122*, 2024a.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yang, J., Jimenez, C., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024b.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of Thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36:11809–11822, 2023.

Yeo, E., Tong, Y., Niu, M., Neubig, G., and Yue, X. Demystifying long chain-of-thought reasoning in LLMs. *arXiv preprint arXiv:2502.03373*, 2025.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Dai, W., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Qiao, M., Wu, Y., and Wang, M. DAPO: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yuan, Y., Yu, Q., Zuo, X., Zhu, R., Xu, W., Chen, J., Wang, C., Fan, T., Du, Z., Wei, X., et al. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025a.

Yuan, Y., Yue, Y., Zhu, R., Fan, T., and Yan, L. What's behind PPO's collapse in long-CoT? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025b.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., Ma, Z., and He, J. SimpleRL-Zoo: Investigating and taming zero

reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025a.

Zeng, Z., Cheng, Q., Yin, Z., Zhou, Y., and Qiu, X. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *arXiv preprint arXiv:2502.12215*, 2025b.

Zhang, X., Wang, J., Cheng, Z., Zhuang, W., Lin, Z., Zhang, M., Wang, S., Cui, Y., Wang, C., Peng, J., et al. SRPO: A cross-domain implementation of large-scale reinforcement learning on LLM. *arXiv preprint arXiv:2504.14286*, 2025a.

Zhang, Z., Zheng, C., Wu, Y., Zhang, B., Lin, R., Yu, B., Liu, D., Zhou, J., and Lin, J. The lessons of developing process reward models in mathematical reasoning. *ArXiv*, abs/2501.07301, 2025b.

Zhao, E., Awasthi, P., and Gollapudi, S. Sample, scrutinize and scale: Effective inference-time search by scaling verification. *arXiv preprint arXiv:2502.01839*, 2025.

Zhu, X., Wang, J., Zhang, L., Zhang, Y., Huang, Y., Gan, R., Zhang, J., and Yang, Y. Solving math word problems via cooperative reasoning induced language models. In *ACL (1)*, pp. 4471–4485. Association for Computational Linguistics, 2023.

A. Related Work

Reinforcement learning with verifiable rewards. Reinforcement learning has shown great promise in improving large language models, as demonstrated by the success of reinforcement learning from human feedback (RLHF) and from AI feedback (RLAIF), which align model responses with human preferences (Lee et al., 2024; Meng et al., 2024; Ouyang et al., 2022; Rafailov et al., 2023). More recently, reinforcement learning with verifiable rewards (Dai et al., 2025; Gao et al., 2024; Havrilla et al., 2024; Lambert et al., 2024a; Liu et al., 2025; Shao et al., 2024; Team et al., 2025b; Wang et al., 2025; Xu et al., 2025a; Yu et al., 2025; Yuan et al., 2025a;b; Zhang et al., 2025a) has attracted growing attention for its effectiveness in incentivizing reasoning in LLMs with rule-based rewards (Fatemi et al., 2025; Guo et al., 2025; Kazemnejad et al., 2024; Team et al., 2025a; Zeng et al., 2025a; Dong et al., 2024; Lambert et al., 2024b; Mroueh, 2025). Notably, DeepSeek-R1 (Guo et al., 2025) and Kimi K1.5 (Team et al., 2025a) demonstrate that RLVR can elicit emergent reasoning behaviors such as long chain of thought and self-reflection, and achieve strong performance across diverse reasoning tasks such as math and coding problems. Yeo et al. (Yeo et al., 2025) further explore the emergence of long CoT across different RLVR setups.

Despite these advances, many prior works mainly focus on evaluating the model's Pass@1 or greedy decoding performance, which might overlook the underlying change in model behavior (*e.g.*, inference scaling performance). More importantly, the mechanisms behind RLVR for driving reasoning and generalization remain underexplored. In this work, we take a step further by decomposing the RLVR objective into two distinct learning paradigms—learning from correct responses and learning from incorrect responses—and analyzing the learning signal at the token level through gradient analysis. We evaluate the models extensively using Pass@k with a wide spectrum of k. Our findings highlight the critical yet previously underappreciated role of negative rewards in RLVR.

Inference-time scaling behaviors. Inference-time scaling has emerged as a promising direction for enhancing model performance (Balachandran et al., 2025; Brown et al., 2024; Chen et al., 2024; Lightman et al., 2024; Muennighoff et al., 2025; Qu et al., 2025; Setlur et al., 2025; Snell et al., 2024; Wang et al., 2024a;b; 2023; Welleck et al., 2024; Wu et al., 2025a; Zhu et al., 2023; Yao et al., 2023; Zelikman et al., 2022; Zhao et al., 2025)—particularly in reasoning tasks where generating multiple candidate solutions (Cobbe et al., 2021; Zhu et al., 2023) or longer reasoning traces (Muennighoff et al., 2025; Yeo et al., 2025; Zeng et al., 2025b) can help hit the correct answer. Through the lens of inference-time scaling, recent studies have raised questions about whether RL-trained models are better than the base models (Yue et al., 2025; Hochlehnert et al., 2025). A recent work by (Yue et al., 2025) suggests that RLVR primarily adjusts the model's output distribution toward high-reward responses, rather than eliciting new reasoning abilities beyond the base model. By adopting Pass@k as metric, they find that RL-trained models have inferior inference-time scaling performance than the base models. These findings align with our findings that reinforcing correct samples hurts Pass@k at larger k. Another concurrent work by (Dang et al., 2025) studies supervised fine-tuning (SFT), and finds that diversity collapse during SFT adversely affects inference-time scaling performance. They show that SFT reduces generation diversity, leading to degraded Pass@k performance, and that interpolating weights from early, potentially less overconfident checkpoints and later ones can effectively restore diversity and improve both Pass@1 and Pass@k.

In this work, we highlight the underestimated role of negative reinforcement in preserving and improving inference-time scaling performance, and that the accuracy and diversity trade-off in RLVR can be effectively balanced by adjusting positive and negative reward weights.

B. Gradient Derivation

Gradient of Equation (6). For simplicity of notation, let $\pi_v \coloneqq \pi_\theta(v|\boldsymbol{x}, y_{<t})$, and we omit the constant $\frac{1}{T}$ in the equation which effectively scales the gradient magnitude.

$$\frac{\partial \mathcal{L}}{\partial z_v} = -R \cdot \frac{\mathbb{1}(v = y_t)\exp(z_{y_t})\sum_{v' \in \mathcal{V}} \exp(z_v) - \exp(z_{y_t})\exp(z_v)}{\left(\sum_{v' \in \mathcal{V}} \exp(z_{v'})\right)^2}$$

For the sampled token $v = y_t$, the gradient simplifies to

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_v} &= -R \cdot \frac{\exp(z_v)\sum_{v' \in \mathcal{V}} \exp(z_{v'}) - \exp(z_v)^2}{\left(\sum_{v' \in \mathcal{V}} \exp(z_{v'})\right)^2} \\ &= -R \cdot \frac{\exp(z_v)\left(\sum_{v' \in \mathcal{V}} \exp(z_{v'}) - \exp(z_v)\right)}{\left(\sum_{v' \in \mathcal{V}} \exp(z_{v'})\right)^2} \\ &= -R \cdot \left(\frac{\exp(z_v)}{\sum_{v' \in \mathcal{V}} \exp(z_{v'})} \cdot \frac{\sum_{v' \in \mathcal{V}} \exp(z_{v'}) - \exp(z_v)}{\sum_{v' \in \mathcal{V}} \exp(z_{v'})}\right) \\ &= -R \cdot \pi_v \cdot (1 - \pi_v)\end{aligned}$$

For the unsampled token $v \neq y_t$, the gradient simplifies to

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_v} &= -R \cdot \frac{-\exp(z_{y_t})\exp(z_v)}{\left(\sum_{v' \in \mathcal{V}} \exp(z_{v'})\right)^2} \\ &= R \cdot \frac{\exp(z_{y_t})}{\sum_{v' \in \mathcal{V}} \exp(z_{v'})} \cdot \frac{\exp(z_v)}{\sum_{v' \in \mathcal{V}} \exp(z_{v'})} \\ &= R \cdot \pi_{y_t} \cdot \pi_v\end{aligned}$$

In summary, the gradient of Equation (6) can be expressed as

$$\frac{\partial \mathcal{L}}{\partial z_v} = \begin{cases} -R \cdot \pi_v \cdot (1 - \pi_v) & \text{if } v = y_t \quad \text{(sampled token)} \\ R \cdot \pi_v \cdot \pi_{y_t} & \text{if } v \neq y_t \quad \text{(unsampled token)} \end{cases}$$

Gradient of entropy regularization. Entropy regularization is also one way to induce a diverse output probability distribution. We compute and analyze its gradient dynamics to reveal its difference from the NSR objective as follows. The entropy loss is

$$\mathcal{H}(\pi_\theta) = -\sum_{v' \in \mathcal{V}} \pi_{v'} \log \pi_{v'},$$

and the gradient ascent direction to maximize entropy is

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial z_v} &= -\sum_{v' \in \mathcal{V}} \frac{\partial \pi_{v'}}{\partial z_v}(1 + \log \pi_{v'}) \\ &= -\sum_{v' \in \mathcal{V}} \pi_{v'} \left(\mathbb{1}(v = v') - \pi_v\right)(1 + \log \pi_{v'}) \\ &= -\left(\pi_v(1 + \log \pi_v) - \pi_v \sum_{v' \in \mathcal{V}} \pi_{v'}(1 + \log \pi_{v'})\right) \\ &= -\pi_v \left(\log \pi_v - \underbrace{\sum_{v' \in \mathcal{V}} \pi_{v'} \log \pi_{v'}}_{=\bar{\ell}}\right)\end{aligned}$$

It can be seen that the gradient sign and magnitude depend on how the log-probability of token v, $\log \pi_v$, compares to the expected log probability $\bar{\ell}$ over the vocabulary: when token v is significantly more probable than expected, it receives a larger negative gradient, pushing its logit down more strongly. In contrast, when token v is less probable than average, it receives a positive gradient that boosts its logit.

As a result, entropy maximization systematically flattens the output distribution by suppressing high-confidence tokens and boosting low-confidence ones. This behavior can conflict with the model's prior knowledge: confidently correct tokens—such as syntactically or semantically appropriate completions—are penalized more for being too probable, while

completely implausible tokens may be promoted simply because they are unlikely under the current policy.

C. RLVR vs. RL with Reward Models

Different from RLVR, many RL setups require training reward models to assign scalar values that indicate the quality of generation (*e.g.*, in RLHF (Ouyang et al., 2022)). During policy optimization, these rewards are often normalized within a batch by subtracting the mean, so that a sample's final reward depends on its relative ranking compared to others in the same batch. As a result, a sample's reward can change sign (from positive to negative or vice versa) depending on the overall quality of the batch, making it difficult to interpret rewards as absolute indicators of positive or negative samples. In contrast, RLVR applies a binary reward ($+1/-1$). The signs of the rewards are inherently tied to the correctness of individual sequences, as each token in a sequence receives the same reward and the batch average reward always remains within $[-1, 1]$, thus reward normalization will preserve the original sign. This inherent sign preservation in RLVR is critical for our analysis, as it enables a clean and unambiguous separation into positive and negative learning paradigms based on the reward signs.

D. Implementation Details

D.1. Objectives and Training Hyperparameters

The objectives of PPO, GRPO, PSR and NSR are as follows:

$$\mathcal{L}_{\text{PPO}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\min\left(\frac{\pi_\theta(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}A_t, \text{clip}\left(\frac{\pi_\theta(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}, 1-\epsilon, 1+\epsilon\right)A_t\right),$$

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T}\sum_{t=1}^{T}\left(\min\left(\frac{\pi_\theta(y_{i,t}|\boldsymbol{x},\boldsymbol{y}_{i,<t})}{\pi_{\text{old}}(y_{i,t}|\boldsymbol{x},\boldsymbol{y}_{i,<t})}A_{i,t}, \text{clip}\left(\frac{\pi_\theta(y_{i,t}|\boldsymbol{x},\boldsymbol{y}_{i,<t})}{\pi_{\text{old}}(y_{i,t}|\boldsymbol{x},\boldsymbol{y}_{i,<t})}, 1-\epsilon, 1+\epsilon\right)A_{i,t}\right) - \beta\text{KL}(\pi_\theta\|\pi_{\text{ref}})\right),$$

$$\mathcal{L}_{\text{PSR}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\min\left(\frac{\pi_\theta(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}, \text{clip}\left(\frac{\pi_\theta(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}, 1-\epsilon, 1+\epsilon\right)\right), \quad r(\boldsymbol{x},\boldsymbol{y}) = 1,$$

$$\mathcal{L}_{\text{NSR}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\min\left(-\frac{\pi_\theta(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}, -\text{clip}\left(\frac{\pi_\theta(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}{\pi_{\text{old}}(y_t|\boldsymbol{x},\boldsymbol{y}_{<t})}, 1-\epsilon, 1+\epsilon\right)\right), \quad r(\boldsymbol{x},\boldsymbol{y}) = -1,$$

where ϵ is the clip ratio, and β is the KL penalty coefficient. For PPO, the KL penalty is applied to the advantage. For GRPO, the KL penalty is added to the final loss. Both PPO and GRPO use a KL penalty coefficient of 1e-3. For PSR and NSR, we do not apply KL penalty, which we find to result in better performance. The learning rate of the critic model in PPO is 1e-5. The clip ratio is set to 0.2. We also apply entropy bonus to all the above objectives, with a coefficient of 1e-4. Our experiments are conducted over a single node with 8 NVIDIA H200 GPUs.

D.2. Prompt Templates

We adopt the prompt templates for Qwen models following (Zeng et al., 2025a), as shown in Tables 2 and 3.

Table 2: Prompt template for `Qwen2.5-Math-7B`.

<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
{input}
Please reason step by step, and put your final answer within \boxed{}.<|im_end|>
<|im_start|>assistant

Table 3: Prompt template for `Qwen3-4B` non-thinking mode.

```
<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
{input}
Please reason step by step, and put your final answer within \boxed{}.<|im_end|>
<|im_start|>assistant
<think>

</think>
```

D.3. Advantage Normalization

The implementation of computing advantage in verl includes an advantage normalization. However, for PSR and NSR, since we exclude the KL penalty, the advantage is equal to the raw reward (*i.e.*, $+1$ for PSR and -1 for NSR). In this case, applying normalization would cause the advantage to be zero, thus eliminating the learning signal. As a result, we disable this normalization in PSR and NSR's implementation. A recent work (Xiong et al., 2025) also suggests that the reward normalization may not be necessary for certain settings.

E. Limitations

We discuss the limitations of our work and future work directions as follows.

1. **Primary focus on Qwen models.** To understand how PSR and NSR interact with and leverage strong model priors, our experiments concentrate on Qwen2.5 and Qwen3 models that are known for their superior reasoning capabilities. As different LMs possess varying levels of prior knowledge and inductive biases, an interesting future direction is to investigate the learning dynamics of PSR, NSR, and W-REINFORCE across a broader range of model families.

2. **Performance degradation with extended NSR training.** We observe that extensive training with NSR (*e.g.*, over hundreds of gradient steps) leads to a noticeable decline in performance, whereas W-REINFORCE does not exhibit this issue. This suggests that NSR's implicit mechanism for preserving prior knowledge may be insufficient to ensure stable training over the long term, and incorporating some degree of PSR may be necessary. Future work could explore more sophisticated methods for integrating NSR and PSR to design more robust RL algorithms, and investigate NSR as a potential warm-up phase before standard RL training given that it improves the full Pass@k spectrum.

3. **Beyond sparse and binary rewards.** In this work, we focus on the RLVR setup with sparse and binary outcome rewards. However, many real-world tasks require dense reward signals that offer more fine-grained feedback (*e.g.*, evaluating intermediate reasoning steps or subjective tasks). It remains an open question how PSR, NSR, or W-REINFORCE would perform in such settings, where the learning signals are continuous rather than discrete and potentially more difficult to interpret.