RESTRAIN: FROM SPURIOUS VOTES TO SIGNALS — SELF-TRAINING RL WITH SELF-PENALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) with human-annotated data has boosted long chain-of-thought (CoT) reasoning in large language models (LLMs), but these gains come at high costs in labeled data while still faltering on harder tasks. A natural next step is experience-driven learning, where models improve without curated labels by adapting to unlabeled data. We introduce REinforcement learning with Self-resTRAINt training (RESTRAIN), a self-penalizing RL framework that transforms the absence of gold labels into a learning signal. Rather than amplifying spurious majority votes, RESTRAIN leverages signals from the model's entire answer distribution, penalizing overconfident rollouts and low-consistent examples while preserving promising reasoning chains. This restraint mechanism integrates seamlessly into policy optimization methods such as GRPO to self-improve without human supervision. On challenging reasoning benchmarks, RESTRAIN delivers large gains using only unlabeled data. On Qwen3-4B-Base and Octo-Thinker Hybrid-8B-Base model, RESTRAIN boosts pass@1 by up to **+140.7% on AIME25**, **+36.2% on MMLU_STEM**, and **+19.6% on GPQA-Diamond**. Remarkably, it comes within 0.4% of a fully supervised counterpart, nearly matching gold-label training while using no gold labels at all. These results demonstrate that RESTRAIN consistently boosts reasoning without supervision.

$$\mathcal{L}_{TTRL}(x;\theta) = -\frac{1}{n}\sum_{i=1}^{n}\min\left(\rho_i A_{i,maj}, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A_{i,maj}\right) - \beta\mathbb{D}_{KL}[\pi_\theta||\pi_{\text{ref}}]$$

$$\mathcal{L}_{RESTRAIN}(x;\theta) = u_x\left[-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} w_j \min\left(\rho_i \hat{A}_{i,j}, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)\hat{A}_{i,j}\right)\right] - \beta\mathbb{D}_{KL}[\pi_\theta||\pi_{\text{ref}}]$$

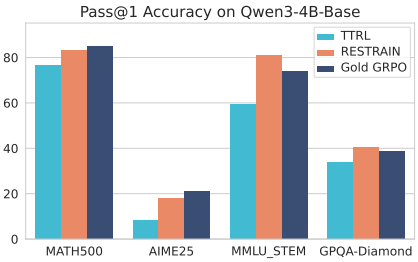
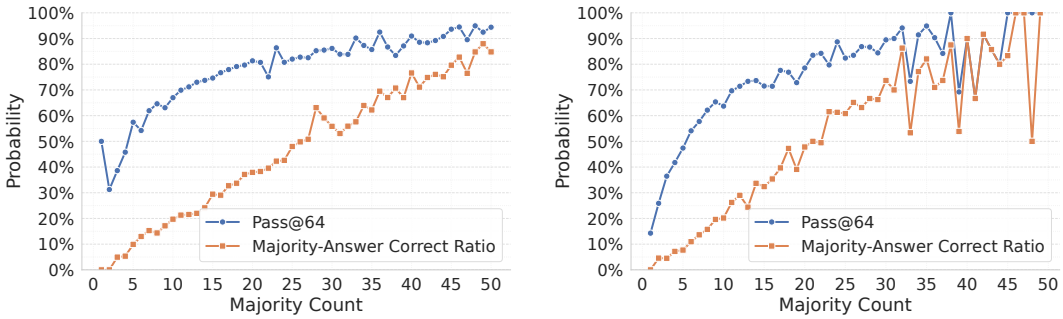


Figure 1: **Self-Training RL.** Our RESTRAIN is a self-penalizing RL method that differs from TTRL by leveraging all distinct answers rather than majority votes, and by introducing label/prompt weighting and advantage penalization for spurious or unreliable rollouts and training examples.

1 INTRODUCTION

Recent advances in LLMs (Guo et al., 2025; Jaech et al., 2024; Yang et al., 2025) show that Reinforcement Learning (RL) with human-annotated data and verifiable rewards (RLVR) greatly enhances long chain-of-thought reasoning (Wei et al., 2022), achieving strong performance on challenging benchmarks. Nevertheless, it often relies on human-designed verifiers and often requires ever-larger amounts of high-quality labeled data. To attain superhuman performance, models will eventually need to operate in environments where even humans lack definitive answers and cannot offer reliable feedback on outputs. In these situations, models must develop the ability to self-improve without direct human supervision. This motivates RL on unlabeled data, where models improve through self-improvement rather than curated labels. Such self-improvement can occur via training with external unlabeled corpora, which broadens accessible data (Zuo et al., 2025). In this paper, we focus on leveraging RL in an unsupervised setting to enhance reasoning generalization.



(a) Qwen3-4B-Base (b) Octothinker Hybrid 8B base

Figure 2: **Evaluation of Majority-Vote Reliability.** We report Pass@64 and the majority-voted accuracy on the DAPO-MATH dataset for Qwen3-4B-Base (left) and OctoThinker Hybrid-8B-Base (right). The large gap between Pass@64 and majority-vote accuracy suggests that correct answers do not always align with majority votes. Moreover, accuracy drops sharply when the majority size is small, indicating that majority votes can carry spurious signals. These observations motivate our self-penalizing scheme, which explores promising reasoning paths beyond unreliable majority votes.

A central challenge in enabling self-improvement without labeled data is how a model can generate its own learning signals. One natural direction is self-rewarding methods, where the model generates its own reward signals—such as ranking or scoring its rollouts based on its own judgment (Yuan et al., 2024). While these methods remove dependence on gold labels, they also risk reward hacking if the self-evaluations reinforce superficial heuristics. The second approach leverages the model's internal properties, such as using majority voting among multiple rollouts to approximate reward signal (Zuo et al., 2025; Shafayat et al., 2025; Liu et al., 2025a). Yet this approach faces reliability and robustness issues, which can cause model training collapse: models frequently generate responses with low self-consistency or low confidence across multiple attempts and for challenging reasoning tasks, the majority-voted answer may be wrong, reflecting systematic reasoning flaws, while minority rollouts can sometimes contain the true solutions(Stahlberg & Byrne, 2019; Stahlberg et al., 2022). Training with overconfident spurious majorities may not provide a reliable reward for handling distorted answer distributions, thus limiting scalability as task diversity and complexity grow. Thus, the key challenge is not simply obtaining any self-derived reward, but ensuring that these self-generated signals foster reliable reasoning improvement.

To address this gap, we introduce RESTRAIN. Instead of relying on gold labels or external supervision, RESTRAIN leverages the model's own predictions by (1) taking into account all models' predicted answers beyond majority-vote, (2) penalizing low-confidence rollouts with negative advantages during policy optimization, (3) discounting prompts with low majority vote. By integrating self-penalization directly into the RL objective, RESTRAIN transforms the absence of labels into rollout-level, prompt-level learning signals. We evaluate RESTRAIN on two base models and two tasks across six benchmarks and observe improvements of up to 20.9% on average. In particular, RESTRAIN raises pass@1 by as much as 140.7% on AIME25, 36.2% on MMLU_STEM, and 19.6% on GPQA-Diamond. Moreover, our method approaches the gold-label supervised model, trailing by 0.4 absolute percentage points. This shows RESTRAIN provides a scalable, unsupervised pathway for reasoning models to self-improve, pushing the reasoning frontier beyond supervised limits.

2 RESTRAIN

We introduce the main ideas of RESTRAIN below and in Figure 3.

Preliminary We adopt Grouped Relative Policy Optimization (GRPO) (Shao et al., 2024) as our main RL algorithm. GRPO optimizes a policy π_θ by sampling n rollouts per prompt x with gold label y, using a value-free, group-mean baseline to reduce variance and a PPO-style clipped surrogate relative to the frozen reference policy π_{ref}. For each rollout y_i, we denote by reward $r_i = R(y_i, y|x)$ with advantage A_i. The GRPO objective is

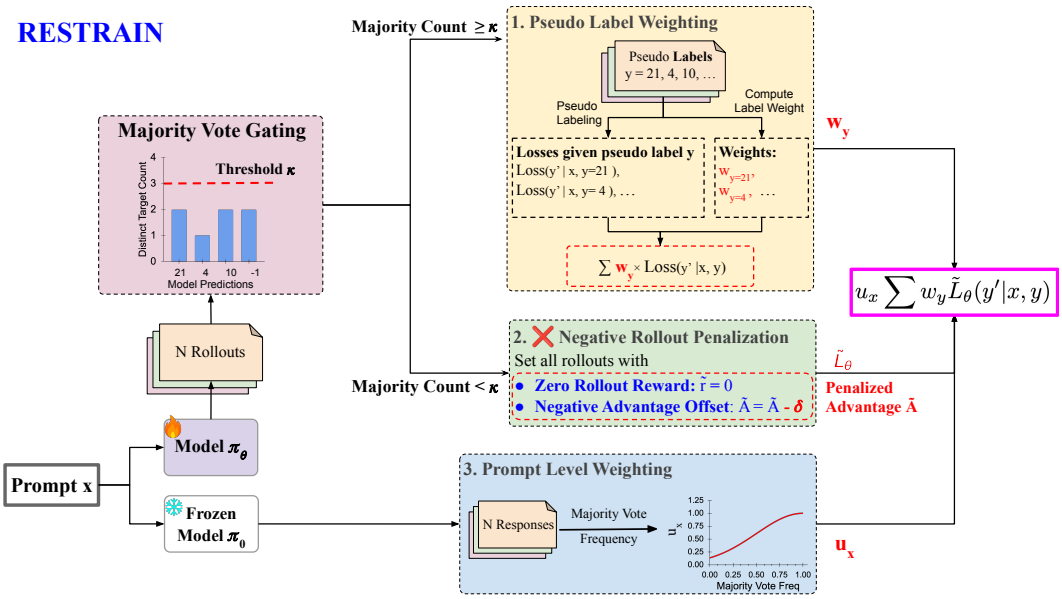


Figure 3: **Overview of Our Method RESTRAIN:** RESTRAIN consists of 3 core components: **1. Label Weighting** which takes into account all possible model-predicted answers as candidate targets when calculating final losses. **2. Negative Rollout Penalization** which penalizes rollouts with very low confidence by setting zero reward and applying negative advantage offsets to the losses. **3. Prompt Level Weighting** which downweighs examples where the model predicts with low self-consistency.

$$\mathcal{L}_{\text{GRPO}}(x, y; \theta) = \frac{1}{n}\sum_{i=1}^{n} \min\Big(\rho_i(\theta)\, A_i,\ \text{clip}\big(\rho_i(\theta), 1-\epsilon, 1+\epsilon\big)\, A_i\Big) \ - \ \beta \mathbb{D}_{KL}\left[\pi_\theta \| \pi_{\text{ref}}\right] \quad (1)$$

2.1 Pseudo-Label Weighting

In unsupervised settings without gold labels, a model might give multiple predictions for a given prompt x, regardless of their correctness. Figure 2 reports majority-vote statistics for the Qwen3-4B-Base model (a) and the OctoThinker Hybrid-8B-Base model (b) on the DAPO datasets. Although the Majority Answer Correct Ratio increases as the number of votes for the majority answer grows, there remains a large gap between Pass@64 and the Majority Answer Correct Ratio, indicating that the correct answer does not always receive the highest number of votes. To address this gap, we introduce a pseudo-label weighting scheme. Rather than collapsing all probability mass onto the single most frequent answer (majority voting) or distributing it uniformly across candidates, our method assigns probabilistic weights proportional to the observed vote counts. This yields a consensus distribution that down-weights spurious low-frequency answers while avoiding the brittleness of requiring unanimity, ensuring each answer is represented according to its empirical frequency.

Construction Given a prompt x, we draw n rollouts $\{y_i\}_{i=1}^{n} \sim \pi_\theta(\cdot \mid x)$ and compute the answer frequencies. Let $\{a_j\}_{j=1}^{m}$ denote the m unique answers extracted from rollouts with counts c_j. and $L_{\text{GRPO}}(x, a_j; \theta)$ be the GRPO loss where a_j is assumed a pseudo label. We then compute the label-free GRPO loss as follows:

$$\mathcal{L}_{\text{GRPO}}(x; \theta) = \sum_{j=1}^{m} w_j \cdot \mathcal{L}_{\text{GRPO}}(x, a_j; \theta) \quad (2)$$

where w_j is a pseudo-label-level weight by applying a monotonic function g to frequency $f_j = \frac{c_j}{n}$:

$$w_j = \frac{g(f_j)}{\sum_{\ell=1}^{m} g(f_\ell)}. \quad (3)$$

Interpretation Equation 3 realizes a *soft selection* over answer frequencies: Model predictions with higher frequencies receive larger probabilities as a pseudo label a_j. The skewness of the weighting is controlled by the parameters of the chosen monotonic shaping function $g(\cdot)$. The choice of

monotonic function g controls how sharply we reward answer frequencies: a steeper g concentrates weight within a narrow range of frequencies, while a smoother parameterizations distribute weight more broadly across answers.

2.2 NEGATIVE ROLLOUT PENALIZATION

Existing methods (Zuo et al., 2025; Shafayat et al., 2025) often rely on the majority-voted answer being correct, making low self-consistency regions prone to unreliable training signals. Our approach instead leverages Pass@n: if any rollout is correct, it provides a valid positive signal, which makes training more robust under weak confidence. However, when the majority count is very low, Pass@n often degrades because the model may generate no correct rollouts at all. As shown in Figure 2, prompts with very low majority counts correspond to unreliable supervision, where no answer can be confidently trusted. To handle such cases, we introduce negative rollout penalization, which assumes all responses to be incorrect and applies a uniform negative offset to all rollouts. This reduces reliance on spurious majorities and encourages the model to explore alternative reasoning paths in low self-consistency regions.

Construction Consider the GRPO loss term $\mathcal{L}_{\mathrm{GRPO}}(x, a_j; \theta)$ associated with pseudo-label a_j. For each rollout y_i, define the reward $r_{i,j} = R(y_i, a_j)$ and the corresponding advantage $A_{i,j}$. Let $M(x) = \max_j c_j$ denote the majority count of prompt x, where c_j is the vote count for label a_j.

When the self-consistency is low ($M(x) < \kappa$), we regard the supervision as unreliable, overwrite the rewards with zero, and apply a uniform offset $\delta \geq 0$ to all such negative rollouts.

$$\tilde{r}_{i,j} = \begin{cases} r_{i,j} & \text{if } M(x) \geq \kappa \\ 0 & \text{if } M(x) < \kappa \end{cases}, \qquad \tilde{A}_{i,j} = \begin{cases} A_{i,j} & \text{if } M(x) \geq \kappa \\ A_{i,j} - \delta & \text{if } M(x) < \kappa \end{cases} \tag{4}$$

In PPO/GRPO objectives, this means that all pseudo labels for prompts with $M(x) < \kappa$ are considered unreliable, and all rollouts are penalized to push the model to avoid spurious low self-consistency reasoning paths.

2.3 PROMPT-LEVEL WEIGHTING

For some prompts, the model exhibits high uncertainty, while for others it produces highly consistent responses. To account for this variation, we scale the update for each prompt by a *fixed* weight that reflects the model's confidence: low-confidence prompts receive smaller updates, and high-confidence prompts receive larger updates. To prevent spurious feedback loops such as artificial confidence inflation during training, these weights are computed once using a frozen base model and kept constant thereafter.

Construction For each prompt x, we sample n rollouts from the reference policy π_{ref} and compute the majority count c_{ref}. Define the prompt weight with a monotonic function $g(\cdot)$:

$$u_x = g(\frac{c_{\text{ref}}}{n}) \tag{5}$$

We apply u_x to each prompt for all subsequent updates. Unlike pseudo-label weights, prompt-level weights are computed offline and then stay fixed throughout the RL training. In Appendix E, we will show offline-computed prompt-level weights outperform online-computed prompt-level weights that are dynamically updated during training.

Final `RESTRAIN` loss Jointly applying label weights w_j from Equation 3 and negative rollout penalization $\tilde{A}_{ij}$ from Equation 4 for each target a_j, and the prompt-level weight u_x from Equation 5, we derived our final `RESTRAIN` loss:

$$\mathcal{L}_{\texttt{RESTRAIN}}(x; \theta) = u_x \sum_{j=1}^{m} w_j \, \tilde{L}_{GRPO}(x, a_j; \theta) \tag{6}$$

$$\Downarrow \text{ expand}$$

$$\tilde{L}_{GRPO}(x, a_j; \theta) = -\tfrac{1}{n} \sum_{i=1}^{n} \min\Big(\rho_i(\theta)\, \tilde{A}_{i,j},\ \mathrm{clip}\big(\rho_i(\theta), 1-\epsilon, 1+\epsilon\big)\, \tilde{A}_{i,j}\Big) - \beta\, \mathbb{D}_{\mathrm{KL}}[\pi_\theta \,\|\, \pi_{\mathrm{ref}}] \tag{7}$$

Table 1: **DAPO-14k-Math:RESTRAIN outperforms all unsupervised baselines.** All results(%) are averaged over 16 seeds. The best results are highlighted in **bold**. RESTRAIN greatly outperforms existing baselines without access to gold label for both Qwen3-4-Base model and Octothinker Hybrid-8B Base models. In particular, RESTRAIN with Qwen4-B-Base without access to gold label reach the performance of GRPO with gold labels.

Model	math.	aime25	olym.	minerva.	mmlu.	gpqa-d.	Avg. ↑
Qwen3-4B-Base	68.0	7.9	35.4	26.0	58.3	32.2	38.0
w/ access to gold labe							
GRPO	85.0	20.8	50.1	40.1	73.7	38.7	51.4
w/o access to gold label							
TTRL	76.3	8.3	39.6	35.9	59.4	33.6	42.2
SRT (easy prompt)	77.8	7.9	39.7	36.3	60.5	34.9	42.8
SRT (offline majority label)	76.9	12.0	39.8	34.2	59.4	34.5	43.1
RESTRAIN (Ours)	**83.0**	**17.9**	**47.0**	**36.5**	**80.9**	**40.2**	**51.0**
Δ(RESTRAIN - TTRL)	+6.7 ↑8.8%	+9.6 ↑115.7%	+7.4 ↑18.7%	+0.6 ↑1.7%	+21.5 ↑36.2%	+6.6 ↑19.6%	+8.8 ↑20.9%
OctoThinker Hybrid-8B-Base	29.8	0.8	12.1	9.3	8.6	24.6	19.2
w/ access to gold label							
GRPO	71.7	6.2	35.2	31.3	62.0	31.0	39.6
w/o access to gold label							
TTRL	56.5	2.7	23.2	22.1	51.7	27.3	30.6
SRT (offline majority label)	58.5	1.7	23.6	27.6	56.4	29.3	32.8
RESTRAIN	**62.1**	**6.5**	**24.0**	26.1	**65.4**	**30.8**	**35.8**
Δ(RESTRAIN - TTRL)	+5.1 ↑9.0%	+3.8 ↑140.7%	+0.8 ↑3.4%	+4.0 ↑18.1%	+13.7 ↑26.5%	+3.5 ↑12.8%	+5.2 ↑17.0%

3 Experimental Setup

Datasets We evaluate the effectiveness of RESTRAIN on two mathematical and reasoning tasks:

- **DAPO-14k-Math**: We adopt the processed DAPO derived from DAPO-Math-17k (Yu et al., 2025b) which deduplicates prompts and standardizes the formatting of both prompts and reference answers. From this release, we further exclude 3k Chinese prompts and use 14k English prompts as our training split, with no further modifications.
- **Synthetic S1k**: A 5k synthetic reasoning dataset from CoT-Self-Instruct (Yu et al., 2025a). Starting from the curated s1k seed set (Muennighoff et al., 2025), Yu et al. (2025a) prompt LLMs to reason step by step and then synthesize new instructions of similar difficulty. Each synthetic example contains both a novel question and a verifiable target answer produced during generation. This dataset complements existing curated math datasets by providing a fully synthetic yet diverse set of reasoning problems, and allows us to systematically test our method under a purely synthetic data generation setting.

Base Models To evaluate the generalizability of our method across different backbone models, we conduct experiments using the following models of various model families and sizes: we use Qwen3-4B-Base model and Octothinker Hybrid 8B base model (Wang et al., 2025b), which is mid-trained from Llama3.1-8B model (Dubey et al., 2024).

Benchmarks Our benchmark suite comprises six publicly available datasets spanning mathematics (four) and science (two). (1) MATH-500 (Hendrycks et al., 2021), (2) AIME25 (Li et al., 2024), (3) OlympiadBench (math subset) (Yang et al., 2024), we use the mathematics portion only. (4) Minerva_math (Yang et al., 2024): the mathematics split from the Minerva quantitative-reasoning suite. (5) MMLU_STEM (Yang et al., 2024), (6) GPQA-Diamond (Yang et al., 2024).

Table 2: **Synthetic S1k dataset: Our RESTRAIN outperforms all unsupervised baselines.** All results(%) are averaged over 16 seeds. The best results are highlighted in **bold**. When training from Qwen3-4B-Base model on synthetic reasoning tasks without using gold label, our method RESTRAIN also outperforms existing unsupervised baselines by 18%.

Model	math.	aime25	olym.	minerva.	mmlu.	gpqa-d.	Avg. ↑
Qwen3-4B-Base	68.0	7.9	35.4	26.0	58.3	32.2	38.0
w/ access to Qwen3-4B label							
GRPO	83.7	18.9	48.4	39.7	83.6	43.5	53.0
w/o access to Qwen3-4B label							
TTRL	76.0	9.2	39.3	35.9	57.6	32.8	41.8
SRT (easy prompt)	76.4	8.1	39.6	34.8	57.5	33.0	41.6
SRT (offline majority label)	75.8	10.4	39.2	33.1	57.1	33.1	41.4
RESTRAIN (Ours)	**81.7**	**20.0**	**45.5**	**36.5**	**73.4**	**40.0**	**49.5**
Δ(RESTRAIN - TTRL)	+5.7 ↑7.5%	+10.8 ↑117.4%	+6.2 ↑15.8%	+0.6 ↑1.7%	+15.8 ↑27.4%	+7.2 ↑22.0%	+7.7 ↑18.4%

Metrics We evaluate with averaged pass@1 (Chen et al., 2021) across six benchmarks, sampling 16 predictions per question using a temperature of 0.6 and a top-p value of 0.95 and averaging their 16 pass@1 accuracy. We use the official evaluation codebase of Qwen2.5-math (Yang et al., 2024).

Baselines We compare RESTRAIN against three recent self-training RLVR methods:

- TTRL (Zuo et al., 2025) treats the majority-voted answer within a group of rollouts as a pseudo ground truth and optimizes the policy to increase its likelihood during RL training.
- Self-Rewarded Training (SRT) (Shafayat et al., 2025) targets the collapse risk of majority-vote supervision with two variants:
  - Offline majority label, an offline majority-vote target computed outside on-policy rollouts, which reduces the tendency to reward self-consistency over correctness;
  - Easy prompts, selecting only "easy" prompts to avoid unstable updates from hard, low-confidence questions. The "easy" prompts are determined by majority vote ratio.
- Entropy-based Test-Time Reinforcement Learning (ETTRL) (Liu et al., 2025a) is an entropy-based strategy that improves test-time reinforcement learning for LLM reasoning. Due to page limit, we put the experimental comparison between RESTRAIN and ETTRL on the Test-Time Training (TTT) task in appendix D.4.

Selection of the monotonic shaping function In our experiments, we compute both the label and prompt-level weights using a Gaussian function centered at the $k \in [0, 1]$ with bias $\sigma > 0$.

4 Main Results

RESTRAIN outperforms unsupervised baselines on two models and datasets On DAPO-MATH-14k Table 1, training without gold labels using RESTRAIN outperforms TTRL and SRT. reaching 51.0%, versus TTRL (42.2%, +8.8 absolute pp), Offline label (43.1%, +7.9 absolute pp), and *Easy prompts* (42.8%, +8.2 absolute pp). A similar pattern holds on the 5k synthetic corpus Table 2, where our method remains the best label-free approach, exceeding the next-best baseline by at least 7.7 absolute pp on average. Moreover, when we exclude the two science benchmarks (MMLU_STEM and GPQA-Diamond), our method is very close to the supervised "reference target" by Qwen3-4B: averaged over the remaining four benchmarks, we obtain 45.9% versus 47.7%—a gap of just 1.8% absolute pp. Similarly, on Octothinker 8B Hybrid Base, we find RESTRAIN has the ability to outperform unsupervised baseline TTRL and SRT by a large amount.

RESTRAIN approximate gold-label upper bound on Qwen3-4B-Base We regard the *Gold-label* setting as an empirical upper bound for label-free RLVR in Table 1 with an average accuracy of 51.4%. Our method is as competitive as GRPO with gold labels, reaching **51.0%** on average—only 0.4 absolute pp below the upper bound. Notably, RESTRAIN surpasses the gold label GRPO, scoring

80.9% vs. 73.7% on MMLU_STEM and **40.2%** vs. 38.7% on GPQA-Diamond, indicating strong generalization without gold-label in these domains. We hypothesize this advantage arises from domain mismatch: the models are trained on DAPO-14k-Math, a math-only dataset, which encourages overfitting to mathematical patterns and slightly degrades transfer to science. In contrast, RESTRAIN leverages distributional signals rather than gold answers, reducing overfitting and preserving cross-domain generalization.

RESTRAIN can effectively prevent model collapse Figure 4 shows the averaged Pass@1 on MATH500 across multiple unsupervised methods. The base model is Qwen3-4B-Base, and all methods are trained on the 14k DAPO dataset. We observe that TTRL improves at first but quickly collapses after 50 steps. In contrast, our method RESTRAIN prevents this sudden collapse and keeps training stable throughout. We attribute this stability to RESTRAIN, which does not exclusively reward the majority-vote answer; instead, it assigns soft weights to all distinct answers in proportion to their empirical frequencies. This frequency-aware weighting smooths the learning signal, curbs overconfident updates, and mitigates sudden collapse.

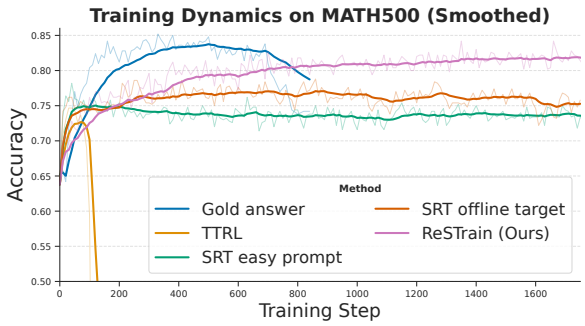


Figure 4: **RESTRAIN has more stable training dynamics.** In contrast to TTRL, our method RESTRAIN steadily improves model performances.

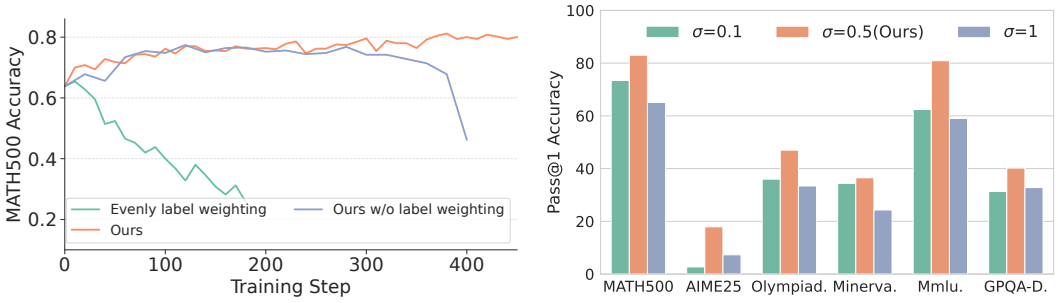
Effectiveness of each component in our RESTRAIN Table 3 presents the impact of each component of our proposed RESTRAIN. The removal of pseudo-label weighting results in the most substantial performance degradation because training collapses quickly. Omitting negative rollout penalization also hurts performance, reducing the average score from 51.0 to 42.1. Finally, removing prompt-level weighting leads to a more modest performance decrease, yet still validates its positive contribution to the model. Taken together, these results show that all components are necessary for stable and effective unsupervised training.

Table 3: **Each component in our RESTRAIN is important.** Each row represents the model's performance with one component removed. The best results are highlighted in bold.

Model	math.	aime25	olym.	minerva.	mmlu.	gpqa-d.	Avg. ↑
RESTRAIN	**83.0**	17.9	**47.0**	36.5	**80.9**	**40.2**	**51.0**
(-) Pseudo-label weighting	67.3	6.0	34.1	24.5	59.3	33.7	37.5
(-) Negative Rollout Penalization	77.3	9.6	39.9	36.2	56.4	33.0	42.1
(-) Prompt-level weighting	82.7	**18.1**	46.7	**37.8**	63.8	37.0	47.7

5 Ablation Study

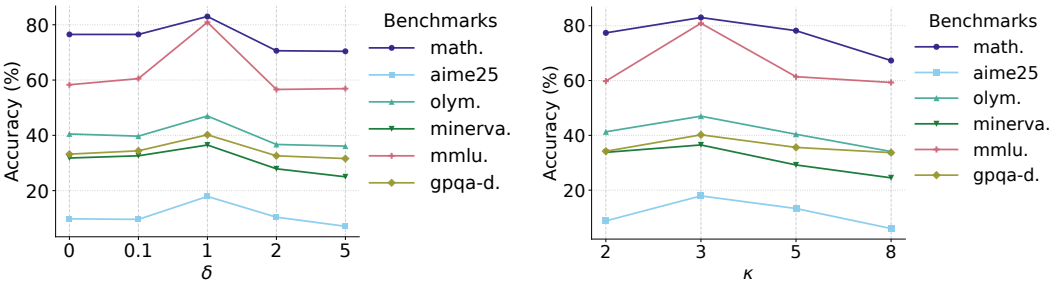
Pseudo-label weighting is crucial to avoid training collapse To assess the impact of our pseudo-label weighting module on training and performance, we run two ablation experiments. One is to apply prompt-level weighting and negative rollout penalization on TTRL, and another is to replace the frequency-based soft weights with uniform weights over all targets for each prompt. Figure 5a reports the outcome: without pseudo-label weighting, training becomes unstable and eventually fails. Uniform weighting performs even worse, accelerating degradation and leading to an earlier collapse. This shows that merely considering all targets is insufficient—low-frequency pseudo-labels are typically erroneous/noisy, and assigning them the same weight as high-frequency (likely correct) pseudo-labels can steer the model in the wrong direction. In contrast, frequency-based soft weighting suppresses rare noise and stabilizes training.



(a) Accuracy curve on MATH500 benchmark.

(b) Performance among different σ value.

Figure 5: **Effect of Pseudo-Label Weighting.** Pseudo-label Weighting prevents training collapse, and the hyperparameter σ can control the "skewness" of the pseudo-label weight distribution.



(a) Performance of our RESTRAIN with different negative advantage offset δ.

(b) Performance of our RESTRAIN with different majority count threshold κ, the number of rollout in our experiment is 16.

Figure 6: **Effect of Pseudo-Label Weighting.** Model performance is sensitive to hyperparameters in Negative Rollout Penalization.

Hyperparameter σ in Pseudo-label Weighting σ controls the "skewness" or concentration of the prompt-level weight distribution. When σ is very small, the weighting approaches a step-like function that sharply distinguishes majority from minority answers, effectively behaving like hard majority voting and largely ignoring less frequent responses. In contrast, a large σ produces a broad, flat distribution, leading to softer, more evenly spread weights across answers. From Figure 5b, a smaller σ ($\sigma = 0.1$) underperforms because it gives too much influence to noisy, infrequent answers. Conversely, a larger σ ($\sigma = 1$) is also suboptimal as it fails to leverage valuable signals from correct minority responses. Thus, $\sigma = 0.5$ provides the best balance, effectively filtering noise while retaining the full distributional signal from the model's outputs.

Hyperparameters in Negative Rollout Penalization Figure 8a ablates the negative advantage offset δ, which dictates the magnitude of the penalty applied to low-consensus rollouts. The results demonstrate that the model's performance is sensitive to δ. With the penalty disabled ($\delta = 0$), the model achieves a similar performance as TTRL, indicating that simply discarding low-confidence prompts does not hinder training. The best accuracy occurs at $\delta = 1$, suggesting that a moderate penalty effectively discourages the model from generating noisy, low-confidence outputs, thereby stabilizing the training signal and enhancing reasoning capabilities. When the penalty magnitude is increased further to δ=2 and δ=5, a consistent and sharp decline in accuracy is observed across all benchmarks. This indicates that an excessively large penalty is detrimental, likely because it over-penalizes the model and may suppress potentially correct, albeit low-frequency, reasoning paths.

Figure 8b varies the majority count threshold κ for triggering the negative penalty; the penalty is applied if the count of the most frequent answer is less than κ. The data reveals a similar trend where performance is suboptimal at both low and high values of κ, peaking at a value of $\kappa = 3$. A threshold that is too lenient ($\kappa = 2$) fails to penalize many noisy, low-confidence training examples, thus limiting performance improvement. Conversely, a threshold that is too strict ($\kappa = 5$ or 8) suppresses

potentially valid reasoning paths in outputs with moderate consensus and causes a significant drop in accuracy. Therefore, the threshold value strikes a crucial balance, effectively filtering unreliable training signals without excessively restricting the model's learning process.

6 Related Work

RL with Verifiable Rewards RL has shown great promise in improving LLMs, as demonstrated by the success of RL from human feedback (RLHF) and from AI feedback (RLAIF), which aligns model responses with human preferences (Lee et al., 2023; Ouyang et al., 2022; Liu et al., 2025b; Yue et al., 2025). More recently, reinforcement learning with verifiable rewards (Gao et al., 2024; Shao et al., 2024; Guo et al., 2025; Yang et al., 2025; Wen et al., 2025; Song et al., 2025; Team et al., 2025; Fatemi et al., 2025; Wang et al., 2025a; Li et al., 2025b) has been developed to further enhance reasoning capabilities in domains such as mathematics and code. Despite its promise, RLVR is largely limited to settings where a verifiable gold label or exhaustive validators exist, and its outcome-based rewards may limit generalization to tasks that are out of distribution.

Unsupervised Reward Estimation Accurately capturing reward signals without relying on human labels has been the focus of several recent studies. Early work like STaR (Zelikman et al., 2022) relies on repeated outcome evaluation. Self-Rewarding LMs (Yuan et al., 2024) explores using LLM-as-a-Judge to provide its own rewards to do self-training. SCPO (Prasad et al., 2024) introduced self-consistency as an alternative to human-annotated rewards, demonstrating its effectiveness in improving reasoning tasks through DPO training. Building on these ideas, TTRL (Zuo et al., 2025) further explored self-consistency signals in an online setting, which treats the majority-voted answer as a pseudo label and leverages the GRPO algorithm (Shao et al., 2024) to update the model. However, TTRL was found to suffer from overconfidence issues, resulting in mode collapse. To address this, SRT (Shafayat et al., 2025) proposed using offline-generated labels and curriculum learning; ETTRL (Liu et al., 2025a) proposed an entropy-based mechanism that enhances the balance between exploration and exploitation, thus mitigating overconfidence and improving overall performance; EVOL-RL (Zhou et al., 2025) introduced novelty reward to increase exploration. Other unsupervised methods derive intrinsic rewards from a model's internal feedback—Reinforcement Learning from Internal Feedback (RLIF). For example, some approaches measure the model's output certainty, using metrics like token- and trajectory-level entropy (Prabhudesai et al., 2025; Agarwal et al., 2025) or self-confidence (Li et al., 2025a). Along these lines, Intuitor (Zhao et al., 2025) utilizes a model's internal confidence termed "self-certainty" as its sole intrinsic reward. Another method, EMPO (Zhang et al., 2025a), uses clustering to extract semantic entropy across multiple rollouts and compute corresponding advantages. Zhang et al. (2025b) theoretically analyzes internal equivalence among RLIF methods and claims that the prior of the base model causes training collapse.

Unlikelihood Penalization Unlikelihood training is a widely adopted technique in neural text generation to penalize undesirable outputs. (Welleck et al., 2019) reduces the probability of specific "negative candidate" tokens. (Li et al., 2019) later employed this approach to improve logical consistency, demonstrating its effectiveness as a general framework for mitigating known biases in dialogue by penalizing a carefully selected set of negative tokens at each generation step. More recently, NSR (Zhu et al., 2025) extended this principle from the neural text generation model to LLMs post-training with their Negative Sampling Rejection (NSR) method. In the context of RLVR, they show that penalizing entire negative trajectories consistently improves performance, preserves generation diversity, and promotes generalization over the base model.

7 Conclusion

In this paper, we propose RESTRAIN, a self-penalizing reinforcement learning framework that transforms the absence of gold labels into a learning signal, enabling models to self-improve without gold labels. By (i) weighting all predicted targets rather than only the majority, (ii) penalizing low-confidence rollouts within the policy objective, and (iii) discounting prompts with low self-consistency, RESTRAIN enables robust self-improvement and mitigates the training collapse of majority-vote heuristics. Empirically, it delivers more stable optimization and stronger generalization on challenging reasoning tasks like math and science.

REFERENCES

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. *arXiv preprint arXiv:1911.03860*, 2019.

Pengyi Li, Matvey Skripkin, Alexander Zubrey, Andrey Kuznetsov, and Ivan Oseledets. Confidence is all you need: Few-shot rl fine-tuning of language models. *arXiv preprint arXiv:2506.06395*, 2025a.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025b.

Jia Liu, ChangYi He, YingQiao Lin, MingMin Yang, FeiYang Shen, ShaoGuo Liu, and TingTing Gao. Ettrl: Balancing exploration and exploitation in llm test-time reinforcement learning via entropy mechanism. *arXiv preprint arXiv:2508.11356*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. Self-consistency preference optimization. *arXiv preprint arXiv:2411.04109*, 2024.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444v1*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mingyang Song, Mao Zheng, Zheng Li, Wenjie Yang, Xuan Luo, Yue Pan, and Feng Zhang. Fastcurl: Curriculum reinforcement learning with progressive context extension for efficient training r1-like reasoning models. *arXiv e-prints*, pp. arXiv–2503, 2025.

Felix Stahlberg and Bill Byrne. On nmt search errors and model errors: Cat got your tongue? *arXiv preprint arXiv:1908.10090*, 2019.

Felix Stahlberg, Ilia Kulikov, and Shankar Kumar. Uncertainty determines the adequacy of the mode and the tractability of decoding in sequence-to-sequence models. *arXiv preprint arXiv:2204.00471*, 2022.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025a.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, et al. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*, 2025.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ping Yu, Jack Lanchantin, Tianlu Wang, Weizhe Yuan, Olga Golovneva, Ilia Kulikov, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. Cot-self-instruct: Building high-quality synthetic prompts for reasoning and non-reasoning tasks. *arXiv preprint arXiv:2507.23751*, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 3, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *arXiv preprint arXiv:2504.05812*, 2025a.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. No free lunch: Rethinking internal feedback for llm reasoning. *arXiv preprint arXiv:2506.17219*, 2025b.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*, 2025.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A A PSEUDO CODE OF THE RESTRAIN LOSS FUNCTION

This section shows a pseudo code of our RESTRAIN loss function calculation for one single prompt.

Listing 1: The pseudo-code of the RESTRAIN loss function for one prompt

```
def restrain_loss(outputs, prompt_weight, threshold, neg_offset):
    # --- Extract answers ---
    answers = [extract_answer(output) for output in outputs]

    # --- Majority size M(x) ---
    counts = Counter(answers)
    Mx = counts.most_common(1)[0][1]

    # ---------------------------------
    # Branch 1: Negative penalization
    # ---------------------------------
    if Mx < threshold:
        rewards = [0.0] * len(outputs)
        adv = calculate_advantages(rewards)
        adv = [a - neg_offset for a in adv]
        loss = calculate_loss(adv)
        return prompt_weight * loss

    # ---------------------------------
    # Branch 2: Pseudo-label weighting
    # ---------------------------------
    # Calculate label weights
    freqs = counts.values() / len(outputs)
    label_weights = calculate_label_weight(freqs)

    # Calculate each label loss, then weighted sum to a final loss
    final_loss = 0.0
    for i, label in enumerate(counts.keys()):
        rewards = [reward_fn(ans, label) for ans in answers]
        adv = calculate_advantages(rewards)
        loss = calculate_loss(adv)
        final_loss += label_weights[i] * loss

    return prompt_weight * final_loss
```

B AN ALGORITHM OF THE PER-PROMPT RESTRAIN LOSS FUNCTION

Algorithm 1: Per-prompt RESTRAIN Loss

Input : Responses $\mathcal{O} = \{o_1, \ldots, o_n\}$; prompt weight $u_x > 0$; majority threshold κ; negative offset $\delta \geq 0$.

Output : Loss L.

$\mathcal{A} = \{a_1, \ldots, a_m\} \leftarrow \mathsf{Set}([\,\mathsf{ExtractAnswer}(o_i)\,]_{i=1}^{n})$; $\quad M(x) \leftarrow \max(\mathsf{Count}(a))$

if $M(x) < \kappa$ **then**

  $r_i \leftarrow 0\ \forall i$; $\quad \mathbf{adv} \leftarrow \mathsf{CalculateAdvantages}(\{r_i\}_{i=1}^{n})$; $\quad adv_i \leftarrow adv_i - \delta\ \forall i$;

  return $L \leftarrow u_x \cdot \mathsf{CalculateLoss}(\mathbf{adv})$

else

  for $j = 1$ **to** m **do**

    $f_j \leftarrow c(t_j)/n$; $\quad \tilde{w}_j \leftarrow \mathsf{CalculateWeight}(f_j)$

  $Z \leftarrow \sum_{j=1}^{m} \tilde{w}_j$; $\quad w_j \leftarrow \tilde{w}_j / Z\ \forall j$;

  $L_{\text{final}} \leftarrow 0$;

  for $j = 1$ **to** m **do**

    $r_i \leftarrow \mathsf{RewardFn}(\mathcal{A}[i],\, a_j)\ \forall i$; $\quad \mathbf{adv} \leftarrow \mathsf{CalculateAdvantages}(\{r_i\}_{i=1}^{n})$;

    $\ell_j \leftarrow \mathsf{CalculateLoss}(\mathbf{adv})$;

    $L_{\text{final}} \leftarrow L_{\text{final}} + w_j \cdot \ell_j$

  return $L \leftarrow u_x \cdot L_{\text{final}}$

C Discussion of Motivation

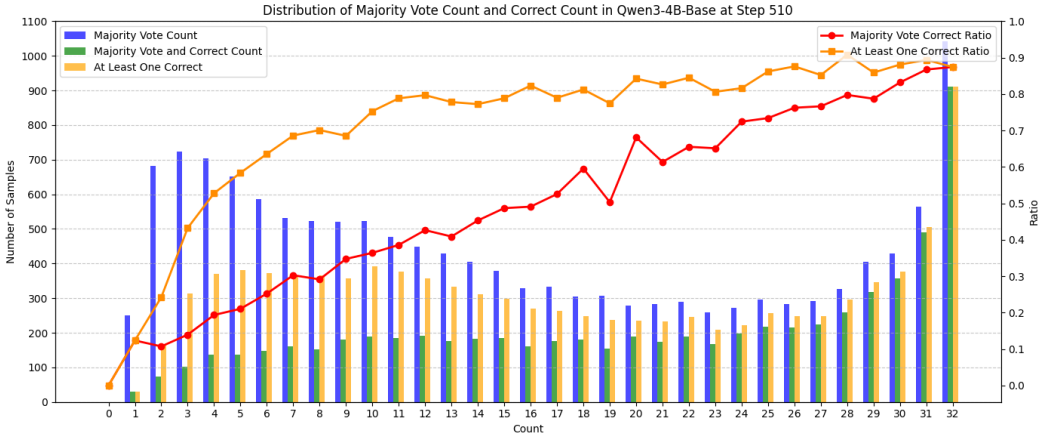


Figure 7: Statistics of Majority Vote Count and Pass@32. The model is trained with Pseudo-label Weighting and Prompt-level weighting. We select a checkpoint when the training converges and use the checkpoint to do inference on the training set to analyze the majority vote count and pass@32.

The figure 7 summarizes majority-vote statistics at step 510 for Qwen3-4B-Base trained with our pseudo-label and prompt-level weighting on the DAPO dataset. The x-axis represents the majority vote count. The chart highlights two key trends: (a) The red line shows the **Majority Vote Correct Ratio**. As the majority vote count decreases (moving left on the graph), the probability that the most frequent answer is actually correct drops almost linearly. (b) The orange line shows the **At Least One Correct Ratio** (i.e. Pass@k). This is the probability that at least one of the generated responses was correct, even if it wasn't the majority answer. This distinction is important for understanding different training methods. A method like TTRL is highly dependent on the majority vote being correct (the red line). When the consensus is low (a low majority vote count), TTRL receives an unreliable and often incorrect training signal. Our proposed method, however, relies on the principle of at least one correct answer being present (the orange line). As long as one of the generated responses is correct, our model receives a valid positive signal for training. This makes it more robust, especially in cases where there isn't a strong consensus on the correct answer. However, the chart also reveals a critical weakness. For very low majority vote counts, the orange line shows a dramatic drop. This indicates that when the model's consensus is extremely low, it's highly probable that none of the generated responses are correct. In this scenario, our method is exposed to significant training noise because there is no positive signal to learn from. To address this specific problem, we introduce our negative rollout penalization to discourage the model from generating sets of answers where none are correct.

D Detailed Results

D.1 Benchmarks

Our benchmark suite comprises six publicly available datasets spanning mathematics (four) and science (two). (1) MATH-500(Hendrycks et al., 2021): a 500-problem subset of the MATH corpus, emphasizing competition-style problems across algebra, geometry, number theory, and combinatorics. (2) AIME25 (Li et al., 2024): the official 2025 American Invitational Mathematics Examination questions. (3) OlympiadBench (math subset) (Yang et al., 2024): olympiad-level problems sourced from national/international contests; we use the mathematics portion only. (4) Minerva_math (Yang et al., 2024): the mathematics split from the Minerva quantitative-reasoning suite. (5) MMLU_STEM (Yang et al., 2024): the STEM categories of MMLU (e.g., physics, chemistry, biology, mathematics-adjacent subjects). (6) GPQA-Diamond (Yang et al., 2024): the highest-difficulty split of GPQA with expert-written, graduate-level science questions spanning physics, chemistry, and biology.

In addition to the 6 benchmarks reported in the main paper, we evaluate on three additional benchmarks. They are (1) **AMC23**(Li et al., 2024): prompts drawn from the 2023 American Mathematics Competitions, covering core high-school problem-solving domains. (2) **AIME24** (Li et al., 2024): the official 2024 American Invitational Mathematics Examination questions. (3) **s1k (verifiable subset)** (Muennighoff et al., 2025): a subset of 893 s1k examples with verifiable answers from Yu et al. (2025a).

D.2 Implementation Details

We implement TTRL, SRT, and RESTRAIN using the VERL codebase. To validate correctness, we reproduce a representative experiment from the original papers with our implementations and verify that the resulting accuracies match. Since ETMR has not released code, we report its results as stated in the original paper. For hyperparameters, we use a learning rate of 1×10^{-6}, and adopt the AdamW optimizer for the policy model. We set kl loss coefficient to 0.001, and the entropy coefficient to 0. For rollout, we sample 16 responses using a temperature of 1.0 for training. The maximum generation length is set to 4096 for Qwen3-4B-Base and Llama3.1-8B-Instruct, and 8192 for Octothinker Hybrid 8B base model. We set the number of epochs to 20. All experiments were conducted on 32 * NVIDIA A100 80GB GPUs.

Table 4: The table shows the evaluation results of training Qwen3-4B-Base on **14k DAPO dataset**, all results(%) are averaged over 16 seeds. The best results are highlighted in **bold**.

Model	math.	amc.	aime24	aime25	olym.	miner.	mmlu.	gpqa.	s1k	avg
Qwen3-4B-Base	68.0	45.6	10.4	7.9	35.4	26.0	58.3	32.2	5.1	32.1
w/ access to gold label										
GRPO	85.0	69.3	21.2	20.8	50.1	40.1	73.7	38.7	12.2	45.7
w/o access to gold label										
TTRL	76.3	52.6	12.0	8.3	39.6	35.9	59.4	33.6	4.6	35.8
SRT (easy prompt)	77.8	52.3	13.5	7.9	39.7	36.3	60.5	34.9	5.6	36.5
SRT (offline majority label)	76.9	51.8	10.4	12.0	39.8	34.2	59.4	34.5	4.7	36.0
RESTRAIN	**83.0**	**60.2**	**20.3**	**17.9**	**47.0**	**36.5**	**80.9**	**40.2**	**10.3**	**44.0**
Oct.Hybrid-8B-Base	29.8	16.1	1.9	0.8	12.1	9.3	8.6	24.6	2.1	15.0
w/ access to gold label										
GRPO	71.7	49.4	10.8	6.2	35.2	31.3	62.0	31.0	7.2	33.9
w/o access to gold label										
TTRL	56.5	32.2	3.9	2.7	23.2	22.1	51.7	27.3	3.5	24.8
RESTRAIN	**61.6**	**33.6**	**6.0**	**8.5**	**24.6**	**25.0**	**64.6**	**29.9**	**4.4**	**28.7**

D.3 Addition Results

Table 4 and 5 show full results of our RESTRAIN on nine benchmarks. Results show that our method can outperform all unsupervised methods on both Qwen3-4B-Base and Octothinker Hybrid 8B base models with two different training datasets.

Table 5: The table shows the evaluation results of training Qwen3-4B-Base on **5k Synthetic S1k dataset**, all results(%) are averaged over 16 seeds. The best results are highlighted in **bold**.

Model	math.	amc.	aime24	aime25	olym.	miner.	mmlu.	gpqa.	s1k	Avg. ↑
Qwen3-4B-Base	68.0	45.6	10.4	7.9	35.4	26.0	58.3	32.2	5.1	32.1
w/ access to Qwen3-4B label										
GRPO	83.7	64.5	23.7	18.9	48.4	39.7	83.5	43.5	11.5	46.4
w/o access to Qwen3-4B label										
TTRL	76.0	50.2	10.8	9.2	39.3	35.9	57.6	32.8	4.8	35.2
SRT (easy prompt)	76.4	52.3	12.1	8.1	39.6	34.8	57.5	33.0	4.9	35.4
SRT (offline majority label)	75.8	53.3	11.9	10.4	39.2	33.1	57.1	33.1	4.6	35.4
RESTRAIN	**81.7**	**58.4**	**17.9**	**20.0**	**45.5**	**36.5**	**73.4**	**40.0**	**8.8**	**42.5**

D.4 RESTRAIN outperforms other Test Time RL Training methods

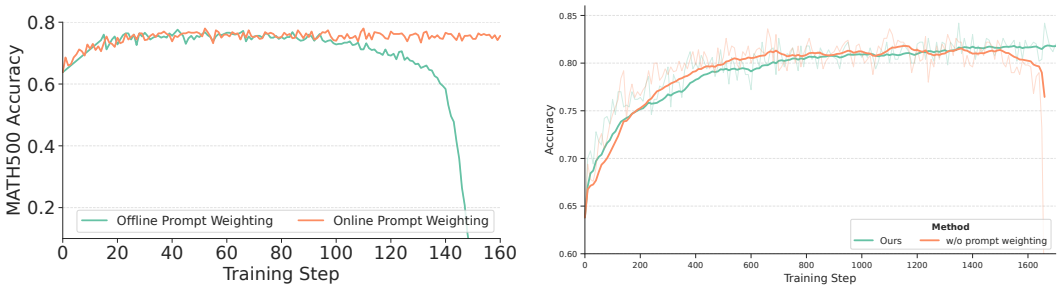
Test Time RL training focuses on the adaptation to test-time data. Compared to recent methods like TTRL (Zuo et al., 2025) and Entropy-fork Tree Majority Rollout (ETMR) proposed by ETTRL (Liu et al., 2025a), our approach achieves consistent improvements across challenging math reasoning benchmarks. As shown in Table 6, RESTRAIN surpasses TTRL and ETMR on AMC23 and MATH-500 by margins of +13.0% and +13.3%, respectively, yielding an overall +11.0% gain in average accuracy without access to gold labels during training. These results demonstrate that our method can also scale more effectively at test time.

Table 6: **Comparing RESTRAIN v.s. Two Test Time RL Training Methods: TTRL and ETMR on Llama3.1-8B-Instruct.** All results(%) are by greedy decoding following Liu et al. (2025a). RESTRAIN also outperforms the existing test-time scaling method by 11%.

Test-Time Method	aime24.	amc	math.	Avg. ↑
TTRL	10.0	32.3	63.7	35.3
ETMR (Liu et al., 2025a)	**16.9**	35.4	59.5	37.3
RESTRAIN (Ours)	16.7	**40.0**	**67.4**	**41.4**
Δ(RESTRAIN - ETMR)	−0.2	+4.6	+7.9	+4.1

E Ablation Study of Prompt-level Weighting

In this section, we evaluate the effect of prompt-level weighting on training. We ablate it by two experiments, one is comparing our offline prompt weighting with online prompt weighting, another is setting all prompt weights to 1 ("w/o prompt weighting"). As shown in Figure 8, in the first experiment, online prompt weighting quickly collapses while offline prompt weighting can continue to improve. For the second experiment, both methods' accuracies improve quickly at the start. Initially, the model without prompt weighting learns slightly faster. However, our method soon overtakes it and consistently maintains a higher accuracy. Notably, the performance of the model without prompt weighting becomes unstable and drops sharply after 1,500 training steps. In contrast, our method's accuracy remains stable and continues to improve. This suggests that offline prompt-level weighting is key to achieving both higher final accuracy and greater training stability.



(a) Online Prompt Weighting collapses very quickly at around 100 steps.

(b) Without Prompt Weighting, the model will ultimately collapse.

Figure 8: **Offline Prompt-weighting can help model train stable.**

F Study of Hyperparameter Weight Bias σ in Pseudo-label Weighting

In this section, we examine the bias parameter σ used in pseudo-label weighting. Table 7 reports the tuning results. When σ is small, the scheme effectively reduces to selecting the majority-vote answer as the pseudo label; when σ is large, it approaches uniform weighting. We observe that σ values near zero or above 1 lead to training collapse and substantially worse performance, whereas a moderate setting (e.g., $\sigma = 0.5$) yields the best stability and accuracy.

Table 7: **Ablation of Pseudo-label Weighting.** The table shows the evaluation results of training Qwen3-4B-Base on **14k DAPO-Math dataset** by varying the hyperparameter weight bias, all results(%) are averaged over 16 seeds. The best results are highlighted in **bold**.

Target Level Weighting Bias σ Small σ = skewed on majority label Large σ = evenly dist. on all labels	math.	aime25	olym.	minerva.	mmlu.	gpqa-d.	Avg. ↑
$\sigma = 0$ (0 weights on non-majority labels)	67.8	7.7	34.7	24.1	58.6	32.1	37.5
$\sigma = 0.1$	73.4	2.7	36.0	34.4	62.4	31.3	40.0
$\sigma = 0.25$	76.5	9.6	39.7	32.6	60.52	34.4	42.2
$\sigma = 0.5$	**83.0**	**17.9**	**47.0**	**36.5**	**80.9**	**40.2**	**51.0**
$\sigma = 1$	65.1	7.3	33.4	24.3	59.0	32.8	37.0
$\sigma = 2$	66.2	6.2	33.1	23.8	58.9	31.4	36.6
$\sigma = 5$	61.1	5.8	32.6	23.7	58.4	33.3	35.8
$\sigma = \infty$ (evenly distributed)	66.8	6.9	34.6	24.6	59.8	32.9	37.6

G Study of Hyperparameter Negative Advantage Offset δ and Majority Count Threshold κ

In this section, we examine how the negative-advantage offset δ and the majority-count threshold κ influence performance. The offset δ scales the penalty applied to low-consensus rollouts; if set too high, it over-penalizes the policy and induces a sharp accuracy decline. The threshold κ decides which prompts are treated as low-consensus: a strict threshold discards many informative examples and hurts accuracy, while an overly loose threshold admits noisy cases and weakens the intended penalization. Appropriate, balanced choices of δ and κ suppress noise without sacrificing useful signal.

Table 8: **Results on Different Negative rollout Penalty.** The table shows the evaluation results of training Qwen3-4B-Base on **14k DAPO-Math dataset** by varying the negative advantage offset, all results(%) are averaged over 16 seeds. The best results are highlighted in **bold**.

Negative Advantage Offset	math.	aime25	olym.	minerva.	mmlu.	gpqa-d.	Avg. ↑
$\delta = 0$	76.5	9.8	40.5	31.8	58.3	33.2	41.7
$\delta = 0.1$	78.7	13.3	40.8	36.5	59.3	35.5	44.0
$\delta = 1$	**83.0**	**17.9**	**47.0**	**36.5**	**80.9**	**40.2**	**51.0**
$\delta = 2$	70.6	10.4	36.7	27.9	56.6	32.6	39.1
$\delta = 5$	70.4	7.1	36.1	25.0	56.9	31.6	37.9

Table 9: **Results on Different Majority Count Threshold.** The table shows the evaluation results of training Qwen3-4B-Base on **14k DAPO-Math dataset** by varying the weight bias, all results(%) are averaged over 16 seeds. The best results are highlighted in **bold**.

Majority Count Threshold (for negative rollouts)	math.	aime25	olym.	minerva.	mmlu.	gpqa-d.	Avg. ↑
$\kappa = 2$	77.4	8.8	41.3	33.8	59.8	34.2	42.5
$\kappa = 3$	**83.0**	**17.9**	**47.0**	**36.5**	**80.9**	**40.2**	**51.0**
$\kappa = 5$	78.2	13.3	40.4	29.2	61.4	35.6	43.0
$\kappa = 8$	67.3	6.0	34.1	24.5	59.3	33.7	37.5

H THE USE OF LARGE LANGUAGE MODELS

In this work, we use LLM for writing polishing and do not use it for any other purpose.