CIPS: CORRECT-INCORRECT PREFIX SCORE AS A BASELINE FOR RL WITH VERIFIABLE REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) is a simple and effective recipe for improving the reasoning capabilities of large language models, but its sequence-level rewards provide only a coarse signal for credit assignment. Cheaper alternatives to process reward models and Monte Carlo sampling use the likelihood of the correct answer as a prefix-level signal, but this likelihood also reflects spurious context and answer syntax rather than reasoning progress alone. We introduce the correct-incorrect prefix score (CIPS), which uses the likelihood of plausible incorrect answers from the same rollout group as a control for these effects. A prefix counts as progress only if it raises the likelihood of correct answers *relative to* incorrect ones. We show that CIPS is robust to spurious prefix content, retains the prompt-difficulty information of the group-mean baseline, and additionally distinguishes rollouts within a group and along the reasoning trace. Used as a prefix-dependent baseline in place of the group mean, CIPS requires neither a learned reward model nor additional rollouts. For two model scales and both mathematical and combinatorial reasoning tasks, CIPS consistently outperforms outcome-only training and prior dense-credit methods. Additionally, with essentially unchanged step time, it reaches the final performance of Dr. GRPO with about 25% shorter training time. The code is publicly available.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a well-established recipe for improving the reasoning capabilities of large language models (LLMs) (Shao et al., 2024; Guo et al., 2025). In its basic form, a model, or *policy*, samples a group of independent answers to a given task, and each answer is verified and assigned a reward. After subtracting the group's mean reward from each reward to form the advantage, an optimization step updates the policy to increase the likelihood of answers with positive advantage and decrease the likelihood of those with negative advantage. This recipe has driven impressive improvements in LLM capabilities in mathematical reasoning, coding, and tool-use tasks (Qwen Team, 2025; Feng et al., 2026a; Wen et al., 2026).

However, rewarding only the final outcome yields a coarse learning signal. A single sequence-level advantage is assigned indiscriminately to every token in a given rollout. Such uniform credit assignment cannot distinguish an individual reasoning step that advances toward a correct solution from one that is redundant or even misleading (Xie et al., 2025a). Likewise, a rollout that fails only because of a small error near its end receives the same signal as one that is wrong from the start.

To provide more granular feedback, process reward models (PRMs) can be trained to evaluate intermediate reasoning steps (Lightman et al., 2024; Wang et al., 2024). PRMs score steps of a reasoning trace and are typically trained on human-annotated or synthetically generated labels. Human supervision is expensive, while automatically generated labels can be noisy and depend on the policy used to produce them (Yuan et al., 2025). Because PRM scores are learned proxies rather than verifiable rewards, policies optimized with them are prone to reward hacking (Gao et al., 2023). Instead of relying on a proxy, Monte Carlo (MC) continuation estimates directly quantify the probability of success from a partial sequence by sampling and evaluating several continuations from the policy (Kazemnejad et al., 2025; Bachmann et al., 2026). While this provides a direct measure of the value of an intermediate state, it comes at substantial computational cost. Sampling multiple continuations at many steps along a sequence can multiply rollout costs by orders of magnitude (Yuan et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

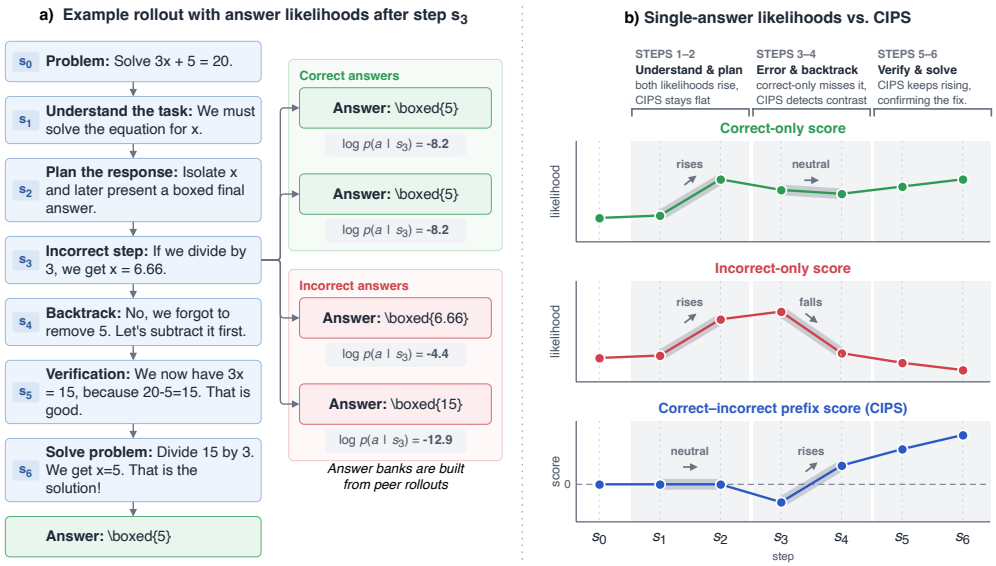


Figure 1: **a)** By calculating the likelihood of answers at intermediate reasoning steps, one can provide step-level credit during reinforcement learning without requiring a learned reward model or additional rollouts. **b)** However, the likelihood of any answer is affected by spurious factors, such as the reasoning chain restating the problem task or planning the answer format. By contrasting the aggregated likelihoods of correct and incorrect answers, our proposed correct-incorrect prefix score (CIPS) is more robust against such factors.

A cheaper alternative evaluates the likelihood of the known correct answer given each reasoning prefix, without sampling any continuations (Xie et al., 2026; Tao et al., 2026). However, the likelihood of immediately emitting the correct answer is not the probability of eventually solving the problem. The former varies with context and answer syntax independently of reasoning progress (Zhao et al., 2021; Kuhn et al., 2023). For example, repeating the required output format or announcing that an answer is imminent raises the likelihood of every plausible answer. Simply using the correct answer's likelihood as a training signal thus conflates reasoning progress with spurious factors.

This work proposes using plausible incorrect answers as a natural control for these spurious factors. A prefix is stronger evidence of progress when it raises the likelihood of correct answers *relative to* incorrect answers for the same prompt. Drawing these incorrect answers from other rollouts in the group does not add any sampling cost and naturally reflects the policy's current failure modes. We operationalize this as the *correct-incorrect prefix score* (CIPS), the difference between aggregated correct- and incorrect-bank likelihoods (Figure 1).

This paper establishes CIPS as an efficient, well-aligned, prefix-dependent baseline for policy optimization. Our contributions are:

- We introduce CIPS as a leave-one-out, prefix-dependent baseline that requires neither a learned reward model nor additional rollouts nor hyperparameter tuning (Section 3).
- We show that the correct-only answer likelihood used in prior work is sensitive to answer readiness and other spurious factors, and thus correlates only weakly with the rollout outcome. CIPS is more robust to such factors, retains prompt-difficulty information, and adds within-group and within-rollout signal (Section 4).
- Across two model scales and both mathematical and combinatorial reasoning tasks, CIPS consistently outperforms outcome-only and prior dense-credit methods with essentially unchanged step time. It improves average math performance over Dr. GRPO (Liu et al., 2025) by 1.6 and 0.9 points for Qwen3-1.7B and Qwen3-4B, respectively, and reaches the final performance of Dr. GRPO with about 25% reduced training time (Section 5).

2 PRELIMINARIES

Policy gradients and baselines. We consider a parametrized policy $\pi_\theta(y \mid x)$ that samples a response $y \sim \pi_\theta(\cdot \mid x)$ given a prompt x. For a data distribution of prompts $x \sim \mathcal{D}$, the goal is to optimize θ to maximize the expected reward

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\ y\sim\pi_\theta(\cdot|x)}[R(x, y)], \tag{1}$$

where $R(x, y)$ denotes a reward function. In our setting, π_θ is an LLM that autoregressively samples a sequence $y_i = (y_{i,1}, \ldots, y_{i,|y_i|})$, before a verifier assigns the completed sequence a binary reward $R_i = R(x, y_i) \in \{0, 1\}$. We write $y_{i,<t} = (y_{i,1}, \ldots, y_{i,t-1})$ to denote the prefix of all tokens preceding position t. Using REINFORCE (Williams, 1992), the gradient of this objective can be estimated as

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\ \{y_i\}_{i=1}^K\sim\pi_\theta(\cdot|x)}\Big[\frac{1}{K}\sum_{i=1}^{K}\sum_{t=1}^{|y_i|}\big(R_i - b_{i,t}\big)\,\nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})\Big], \tag{2}$$

where $b_{i,t}$ is an action-independent baseline. Admissible baselines leave the expected policy gradient unchanged while potentially reducing its variance (Williams, 1992; Greensmith et al., 2004). Intuitively, a good baseline estimates the expected reward given the prefix, $\mathbb{E}[R_i \mid x, y_{i,<t}]$, so that $R_i - b_{i,t}$ reflects only how much better or worse the sampled continuation did than expected. A candidate baseline's correlation with the sequence reward is, therefore, a useful diagnostic of its quality.

Outcome-only reinforcement learning with verifiable rewards. Group-relative policy optimization (GRPO) (Shao et al., 2024) instantiates Equation 2 with a simple baseline. For prompt x, the policy samples K conditionally independent rollouts $y_1, \ldots, y_K$. The baseline is the group mean $\bar{R} = \frac{1}{K}\sum_{i=1}^{K} R_i$, a single scalar per prompt, and every token of rollout y_i receives the advantage $A_i = R_i - \bar{R}$. As proposed in DAPO (Yu et al., 2025), groups with an average reward of zero or one are filtered out, keeping the number of gradient-contributing prompts stable across batches.

Reasoning boundaries and answer likelihood. GRPO's baseline corrects for the varying difficulty of individual prompts. However, every token in a rollout receives the same advantage, independent of its contribution. To obtain a more localized learning signal, rollouts can be partitioned at reasoning boundaries, such as paragraph breaks, and the resulting segments can be evaluated independently (Lightman et al., 2024). Let $s_{i,k}$ denote rollout y_i's prefix ending at the k-th boundary. An efficient way to score each prefix is to measure how strongly it supports a known answer $a = (a_1, \ldots, a_{|a|})$. Given a scorer π_ϕ, which can be the policy itself ($\phi = \theta$) or a separate model, we score a prefix by the length-normalized log-likelihood of the answer tokens,

$$\ell_\phi(a \mid s_{i,k}) = \frac{1}{|a|}\log \pi_\phi(a \mid x, s_{i,k}). \tag{3}$$

Prior work uses the likelihood of the reference answer $a^\star$, $\ell_\phi(a^\star \mid s_{i,k})$, as a step-level reward or shaping term (Xie et al., 2026; Tao et al., 2026). Depending on the implementation, the likelihood of the answer's template opening τ (e.g. `\boxed{`) is scored explicitly, $\ell_\phi(\tau, a^\star \mid s_{i,k})$, as *full-target* score, or the template tokens are provided as part of the context, $\ell_\phi(a^\star \mid s_{i,k}, \tau)$, as *within-template* score.

While either *correct-only score* is cheap to compute, it is not a success probability. It is sensitive to any content that makes returning an answer more likely, such as the prefix repeating the output format or announcing that an answer will follow next, as we show in Section 4.

3 CORRECT-INCORRECT PREFIX SCORE

As established in the previous section, spurious factors in the prefix influence the likelihood of *any* answer, correct or not. Plausible incorrect answers are already available at no extra sampling cost from failed rollouts in group-relative training. This work proposes using their likelihood as a control for that of correct answers, so one can disentangle the signal related to eventual reasoning success from unrelated spurious factors. A step thus counts as progress only if it raises the likelihood of correct answers *relative to* that of incorrect ones.

Peer groups and answer banks. For a target i, let $P_i = \{j \in \{1, \dots, K\} : j \neq i\}$ be its peer set. From peer answers we construct the correct and incorrect answer banks

$$\mathcal{A}_i^+ = \{a^\star\} \uplus \{a_j : j \in P_i,\ R_j = 1\}, \quad \text{and} \quad \mathcal{A}_i^- = \{a_j : j \in P_i,\ R_j = 0\}, \tag{4}$$

where $a^\star$ is a reference answer (the dataset-provided ground-truth answer) and $\uplus$ denotes multiset union. The target's own answer is not part of either bank. Duplicates are retained so recurring policy answers receive proportionally more probability mass. Because $\mathcal{A}_i^-$ is drawn from the current group, it tracks the policy's current failure modes.

Correct-incorrect prefix score. For prefix $s_{i,k}$, we aggregate the answer scores from Equation 3 within each bank and take their difference to compute the *Correct-Incorrect Prefix Score* (CIPS) as

$$\Phi_i(k) = \mathrm{LME}_{a \in \mathcal{A}_i^+} \ell_\phi(a \mid s_{i,k}) - \mathrm{LME}_{a \in \mathcal{A}_i^-} \ell_\phi(a \mid s_{i,k}), \tag{5}$$

where $\mathrm{LME}_{a \in \mathcal{A}} z_a = \log\left(|\mathcal{A}|^{-1} \sum_{a \in \mathcal{A}} e^{z_a}\right)$ aggregates a bank in likelihood space, so that most likely bank entries have a larger impact while retaining a signal from the rest. A larger CIPS value indicates that the prefix favors correct answers more strongly *relative to* incorrect ones.

CIPS-based advantage. At each boundary k, we replace the group-mean baseline of GRPO with a scaled $\Phi_i(k)$, computed from the prefix $s_{i,k}$ and peer answers. Every token in the subsequent segment receives the advantage

$$A_{i,k} = R_i - c\,\Phi_i(k), \tag{6}$$

where c is the lagged scalar coefficient defined below. The baseline remains fixed for the segment k and is recalculated at the next boundary. If the target rollout is the group's only incorrect rollout, its leave-one-out incorrect bank is empty. For this target, we use the Dr. GRPO group-mean advantage.

Weighting of CIPS baseline. The coefficient c matches the scale of the CIPS score to that of the reward: while $R_i \in \{0, 1\}$, we typically observe $-20 < \Phi_i(k) < 20$ during training. Rather than manually tuning c for each model and dataset, we compute it dynamically online. For group g, we pool all rollouts i and prefix positions k, compute a per-group estimate $\hat{c}_g$, and fold it into an exponential moving average c_g:

$$\hat{c}_g = \mathrm{clip}\left(\frac{\sum_{i,k} \Phi_i(k) R_i}{\sum_{i,k} \Phi_i(k)^2 + \varepsilon}, 0, 1\right), \quad \text{and} \quad c_g = \beta\, c_{g-1} + (1 - \beta)\, \hat{c}_g. \tag{7}$$

This fraction is the scale that best fits Φ to R in the squared-error sense, and ε ensures numerical stability. The moving average with decay rate β, set to 0.95, smooths the noisy per-group estimates. In practice, we find that $\hat{c}_g$ is fairly stable, so the choice of β is not critical. Group g uses the previous coefficient c_{g-1}, and c_g is updated only after the group is processed, so a rollout does not use a coefficient fitted on its own data. We note that Venkatraman et al. (2026) concurrently and independently proposed a similar online-fitted coefficient, though for a different mechanism (a privileged value function).

Admissibility as unbiased baseline. Under independent on-policy sampling, the peer banks are independent of the target rollout, conditional on the prompt. For any target token in the segment following boundary k, $c\,\Phi_i(k)$ depends only on these banks and a preceding prefix. Its expected product with the token's score function is therefore zero, so the CIPS baseline preserves the expected REINFORCE gradient for any fixed or lagged c (Williams, 1992; Ahmadian et al., 2024). The full proof is in Appendix A. PPO-style clipping and conditioning training on mixed-outcome groups, however, generally bias the gradient for $\mathcal{J}$ and can shift the effective optimum and training dynamics. These choices are shared with other methods and not introduced by CIPS.

4 Comparison of CIPS to other baseline signals

CIPS is motivated by the observation that spurious prefix content shifts the likelihood of correct and incorrect answers alike, so their contrast should track reasoning progress more faithfully than either likelihood alone. Beyond this robustness, a good baseline should estimate the expected reward given the prefix, $\mathbb{E}[R(x, y_i) \mid x, y_{i,<t}]$, as discussed in Section 2. Before using CIPS for training, we

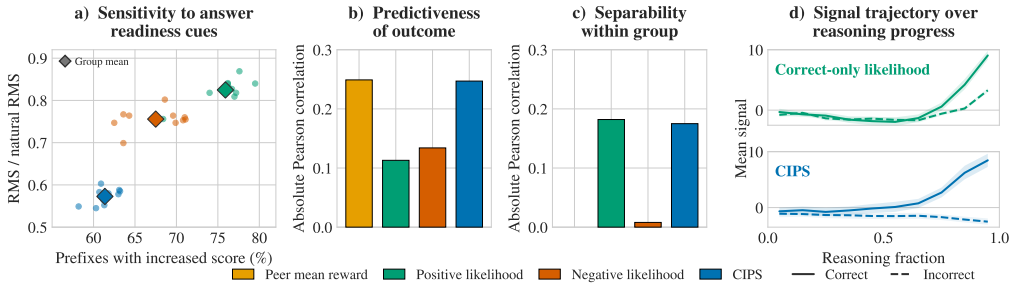


Figure 2: Comparison of CIPS with the peer mean reward and the correct-only and incorrect-only likelihoods as prefix baselines. **a)** Effect of inserting sentences into the rollout that signal answer readiness without advancing the reasoning. Each marker is one of ten tested sentences, and diamonds mark the mean. **b) & c)** Absolute Pearson correlation with the final and the group-centered reward. **d)** Mean signal over reasoning progress, split by final outcome. CIPS separates correct and incorrect rollouts before the midpoint, while the correct-only likelihood separates them only near the end.

therefore test these properties and the ability of CIPS to track the expected reward in a static analysis. We compare it against the peer mean reward used in GRPO, the correct-only likelihood used in prior work, and the incorrect-only likelihood, which forms the second component of CIPS. Specifically, we test whether each signal is robust to prefix content that does not advance the solution, captures prompt difficulty, distinguishes rollouts for the same prompt, and separates successful from failed rollouts early along the trace.

Experimental setup. We collect rollout groups for 1,024 prompts from DAPO-Math-17k (Yu et al., 2025) at three checkpoints of an outcome-only Dr. GRPO run and split each rollout at paragraph boundaries. From each rollout, we select the boundaries closest to 25%, 50%, and 75% to be included in this analysis. We compare CIPS with the correct-only score, the incorrect-only score, and the leave-one-out peer reward rate $\bar{R}_{-i} = \frac{1}{K-1}\sum_{j\neq i} R_j$, which excludes the target rollout so that its own reward R_i does not inflate the correlation by construction. For each single-answer likelihood, we use the within-template likelihood as defined in Section 2, since it performs much better than the full-target likelihood (Appendix C.4).

Robustness to spurious prefix content. To measure robustness to prefix content that does not advance the solution, we append one of ten sentences to each prefix, such as "Now we can answer" or a reminder to use the boxed-answer format, and measure the resulting change in each signal. None of these sentences contains answer-specific information, and continued sampling shows that roughly 90% of prefixes retain their outcome after inserting the sentence (see Appendix B for the full list of sentences). Since the signals differ in scale, we report the root-mean-square (RMS) change caused by an insertion relative to the RMS change of the same signal over a natural reasoning step, i.e., between consecutive boundaries of the unperturbed rollout. We also track how often an insertion increases the signal. The inserted sentences increase the correct-only and incorrect-only scores in 76% and 68% of cases, respectively, and shift these scores by an average of 82% and 76% of the rollouts total RMS (Figure 2a). CIPS is not completely insensitive, but comparably more robust, increasing in only 60% of cases after inserting the sentence and with an average shift of 57%.

Predictiveness of final reward. A baseline should contain information about the final outcome, in our case, the rollout reward R. The peer mean reward shows a correlation of 0.249 with the final reward, as it captures prompt difficulty (Figure 2b). CIPS reaches a similar correlation of 0.247, while the correct-only and incorrect-only scores have absolute correlations of just 0.113 and 0.134, respectively. Contrasting the two banks, therefore, retains more of the prompt-difficulty information in the peer reward rate than either single-answer likelihood alone.

Within-group discrimination. In addition to prompt difficulty, a baseline should be able to differentiate between rollouts within a group. The peer mean reward is not designed for this and is therefore omitted here. To measure the ability to separate rollouts for a given prompt, we center all signals within each group and measure their correlation with $G_i = R_i - \bar{R}$. CIPS and the correct-

Table 1: Performance comparison on math benchmarks, reported as pass@1 and pass@k for Qwen3-1.7B and Qwen3-4B. Values are means over three training seeds ± standard error. **Bold** indicates the highest mean, underline indicates the second-highest. Rows marked with Δ_{baseline} display point differences relative to Dr. GRPO.

	MATH-500		AIME		Minerva		
Method	**p@1**	**p@4**	**p@1**	**p@8**	**p@1**	**p@8**	**Avg**
Qwen3-1.7B	31.0 ±0.6	60.9 ±0.2	0.8 ±0.0	4.4 ±0.3	9.5 ±0.1	31.5 ±0.1	23.0 ±0.0
Dr. GRPO	59.1 ±0.3	77.9 ±0.4	3.5 ±0.2	13.2 ±1.0	21.7 ±0.2	42.7 ±0.3	36.4 ±0.2
VinePPO	45.7 ±1.0	71.7 ±0.1	2.0 ±0.2	9.0 ±2.9	15.2 ±0.6	38.8 ±0.6	30.4 ±0.1
TIPS	61.1 ±0.0	**78.1** ±0.1	4.4 ±0.4	13.3 ±0.3	22.9 ±0.1	44.0 ±0.5	37.3 ±0.0
CIPS	**61.5** ±0.1	**78.1** ±0.1	**4.5** ±0.3	**15.8** ±0.9	**23.8** ±0.3	**44.1** ±0.3	**38.0** ±0.2
Δ_{baseline}	+2.4	+0.2	+1.0	+2.6	+2.1	+1.4	+1.6
Qwen3-4B	38.5 ±0.3	74.1 ±0.2	4.1 ±0.2	17.8 ±0.1	18.0 ±0.2	46.8 ±0.9	33.2 ±0.2
Dr. GRPO	75.0 ±0.6	**88.2** ±0.4	10.2 ±0.5	25.1 ±2.4	33.9 ±0.2	54.0 ±0.5	47.7 ±0.7
VinePPO	54.0 ±0.4	82.0 ±0.8	6.7 ±0.2	21.9 ±0.6	23.5 ±0.1	50.5 ±0.3	39.8 ±0.2
TIPS	74.9 ±0.3	87.5 ±0.1	10.8 ±0.5	23.9 ±1.2	34.3 ±0.4	53.8 ±0.3	47.5 ±0.2
CIPS	**76.2** ±0.4	88.1 ±0.2	**11.0** ±0.5	**26.6** ±1.3	**35.3** ±0.1	**54.6** ±0.4	**48.6** ±0.3
Δ_{baseline}	+1.2	-0.1	+0.8	+1.5	+1.4	+0.6	+0.9

only answer likelihood show a tangible correlation with the group-centered reward, with 0.175 and 0.182, respectively (Figure 2c), while the incorrect-only score only achieves a correlation of 0.027.

Separation along the trajectory. Compared to the group-mean reward, prefix baselines are able to assign individual scores to the segments of a rollout. Naturally, answer likelihoods increase late in the rollout as the reasoning comes to a conclusion. However, ideally, the likelihood trajectories would differ for rollouts depending on whether they ultimately end up being correct or incorrect. We find that correct-only likelihood separates successful and failed rollouts only in the final fifth of the trace (Figure 2d), whereas CIPS separates them already after 40% of the trace. CIPS thus provides a signal that is aligned with the eventual outcome considerably earlier in the trace than the correct-only likelihood.

Across these tests, CIPS is the only signal that is simultaneously robust to spurious cues, captures prompt difficulty, ranks rollouts within a prompt, and separates outcomes early along the trace.

5 Experiments and Results

Next, we test whether using CIPS as a baseline improves the performance compared to outcome-only and other prefix-based methods, on math and combinatorics tasks.

Models and tasks. For our experiments, we train Qwen3-1.7B-Base and Qwen3-4B-Base models (Qwen Team, 2025). We first test mathematical problem solving by fine-tuning on DAPO-Math-17k-Processed (Yu et al., 2025) and evaluating on MATH-500, Minerva Math, and AIME (combined 2024 and 2025) (Hendrycks et al., 2021; Lewkowycz et al., 2022). Our second task is Countdown (Gandhi et al., 2024), in which the model must combine a given set of numbers to obtain a target value using basic arithmetic operators. We adapt task difficulty to model size by varying the number of input numbers. For all experiments, we use three seeds and report the mean final performance.

Compared methods. We compare CIPS to Dr. GRPO, VinePPO, and TIPS. Dr. GRPO (Liu et al., 2025) improves GRPO (Shao et al., 2024) by removing its length bias and the division by the group's reward standard deviation. VinePPO (Kazemnejad et al., 2025) estimates prefix values by Monte Carlo sampling of continuations from each step, which is accurate but computationally very expensive. TIPS (Xie et al., 2026) uses the change in correct-answer likelihood between adjacent steps as reward shaping. All methods share sampling config, optimizer, and the Dr. GRPO surrogate, which

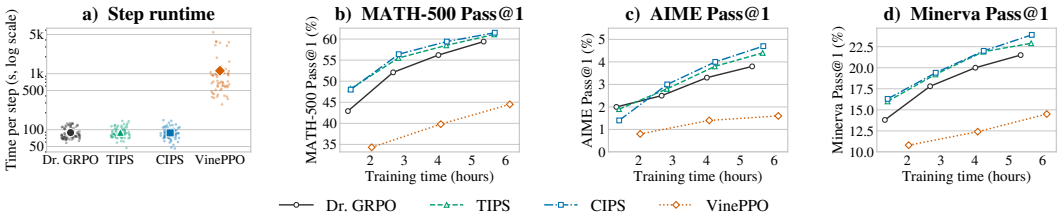


Figure 3: Training cost and efficiency of CIPS compared to the baselines. **a)** Time per training step, measured over three runs of 30 steps on identical prompts. Small markers show individual steps, and large markers show the mean. **b), c) & d)** Pass@1 on MATH-500, AIME, and Minerva over training time, averaged over three seeds.

sums over all tokens of a batch $\mathcal{B}$ of rollouts sampled from $\pi_{\theta_{\text{old}}}$, with x_i the prompt of rollout y_i,

$$\widehat{\mathcal{J}}(\theta) = \frac{1}{Z} \sum_{i \in \mathcal{B}} \sum_{t=1}^{|y_i|} \min\Big(r_{i,t}(\theta)\, A_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right) A_{i,t} \Big), \tag{8}$$

where $r_{i,t}(\theta) = \pi_\theta(y_{i,t} \mid x_i, y_{i,<t}) / \pi_{\theta_{\text{old}}}(y_{i,t} \mid x_i, y_{i,<t})$ is the token-level importance ratio, ϵ the clipping range, and Z a constant normalizer independent of rollout lengths (Liu et al., 2025). Each method supplies its own advantage $A_{i,t}$. For CIPS, the initial policy serves as a fixed scorer, all scoring passes are detached from optimization, and scorer parameters stay fixed within a batch. We train for 240 steps in the main evaluation and 120 steps in ablations, except for VinePPO, whose number of steps we reduce to match the time budget of the other runs. For TIPS, we also tested within-template scoring and the CIPS contrast in place of the correct-only likelihood, and report the best variant. Further details are in Appendix C.

Performance. Table 1 reports pass@1 and pass@k, with $k = 4$ for MATH-500 owing to its size and $k = 8$ otherwise (Chen et al., 2021), averaged over three seeds. For Qwen3-1.7B, CIPS outperforms Dr. GRPO on every benchmark, for both pass@1 and pass@k, raising the average performance from 36.4 to 38.0 (1.6 points, +4.4%), reaching GRPO's performance in about 25% less time (Figure 3b,c,d). It also outperforms the other prefix-based methods, VinePPO and TIPS, on every metric except for a tie with TIPS on MATH-500 pass@4. TIPS is the strongest competitor, but the improvement of CIPS is still 0.7 points on average. VinePPO trails Dr. GRPO throughout, as its high step time permits only a fraction of the training steps within the fixed time budget.

Experiments using Qwen3-4B yield similar results. CIPS achieves the best score on five of six metrics. Dr. GRPO performs better than TIPS on average and four of six metrics, but still trails behind CIPS consistently and on average by 0.9 points (+1.9%), with larger gaps on the pass@1 metrics. Further results, including statistical analysis, per-seed results, and learning rate sensitivity, are presented in Appendix C.1.

The results on a second task, Countdown, follow the same pattern as for mathematical reasoning. CIPS shows the strongest performance across all metrics and for both model sizes, averaged over three seeds. The full evaluation is presented in Appendix C.3.

Runtime overhead. The runtime of an RL method is determined by two factors: the time of each training step and the number of tokens generated during rollouts. We measure the step time by running each method three times for 30 steps on identical prompts, and vary the order of runs to reduce external effects (Figure 3a). As TIPS and CIPS only add inexpensive likelihood evaluations on top of Dr. GRPO, their step times are almost identical, whereas VinePPO takes an order of magnitude longer per step to sample its continuations. Since step times match, the performance gains of CIPS translate directly into wall-clock savings. CIPS leads throughout training and reaches the final performance of Dr. GRPO with about 25% less training time (Figure 3b,c). Beyond the update step itself, a large share of the runtime is spent sampling rollouts, so a method that changes the completion length affects the total cost even when its step time remains unchanged. Over the full Qwen3-1.7B training runs, CIPS generates slightly fewer tokens than both baselines (113.2M per run vs. 113.6M for Dr. GRPO and 114.6M for TIPS), while VinePPO samples 187.2M tokens due to the repeated continuations. See Appendix C.5 for details on memory overhead.

Table 2: Ablations on MATH-500 and AIME, trained for 120 steps, three seeds each. Each block changes one component of the **default** CIPS configuration.

	MATH-500		AIME			MATH-500		AIME	
Configuration	**p@1**	**p@4**	**p@1**	**p@8**	**Configuration**	**p@1**	**p@4**	**p@1**	**p@8**
Default (Ours)	**57.4**	**77.4**	**3.5**	**13.8**	**Default (Ours)**	**57.4**	**77.4**	**3.5**	**13.8**
Dr. GRPO	54.8	76.2	3.1	10.3	Dr. GRPO	54.8	76.2	3.1	10.3
a) Correct-only Supervision					*d) Scorer Update Interval*				
Within template	57.2	76.4	3.5	13.0	Moving ($m = 0$)	57.7	76.9	2.9	11.1
Full target	57.2	77.2	2.5	10.0	$m = 5$	57.5	76.5	3.1	11.9
b) Answer Bank Aggregation					*e) Scorer Scaling*				
Max	58.0	77.1	2.9	12.7	1.7B $\leftarrow$ 1.7B	57.4	77.4	3.5	13.8
Mean	56.5	77.2	2.9	11.8	1.7B $\leftarrow$ 4B	57.2	77.0	3.4	14.0
c) Incorrect Answer Bank					4B $\leftarrow$ 1.7B	72.6	87.2	9.4	25.4
Unrelated	56.2	75.9	3.2	11.8	4B $\leftarrow$ 4B	72.4	87.0	9.8	26.6

Answer-bank construction. In a series of ablations, we change one component of CIPS answer-bank construction at a time and evaluate its impact on the performance of Qwen3-1.7B for MATH-500 and AIME. First, we ablate the incorrect likelihood control with full-target and within-template scoring (Table 2a). Both, however, perform worse than the CIPS formulation, validating the utility of incorrect likelihoods as control. In a second ablation experiment, we test different aggregations for the likelihoods of the banks (Table 2b). Both max and mean aggregations perform worse overall than the default log-mean-exp construction. Finally, to test if the incorrect bank has to contain plausible incorrect answers or if the control also works with unrelated incorrect answers, we shuffle the incorrect banks between prompts (Table 2c). We see that this leads to a substantial drop in performance, even though it would still outperform outcome-only Dr. GRPO in three of four metrics.

Scorer stability and capacity. In a second set of ablations, we test two design choices related to the CIPS scorer. First, we compare the default fixed base-model scorer with dynamic scorers that are updated every m steps, including, in the extreme, $m = 0$, where the trained policy scores its own rollouts (Table 2d). We find that the fixed scorer performs best in three of four metrics. Replacing the scorer with a smaller or larger base model changes performance surprisingly little, motivating more computationally efficient scorers (Table 2e).

Advantage scale and shift. Compared to the group mean, the CIPS baseline is small on average, which shifts and rescales the advantages relative to Dr. GRPO. To test whether this rescaling alone explains the improvement, we train Dr. GRPO with advantages scaled to match the magnitude of CIPS' advantages, once with a constant offset and scale and once with an online rescaling. Both variants improve over Dr. GRPO but remain below CIPS (Appendix C.4). Similarly, CIPS leads at every learning rate we tested (Appendix C.4). The advantage scale thus explains part of the improvement, while the remaining improvements most likely stem from the prefix-dependent baseline.

6 Related Work

Reinforcement learning for large language models. RLVR has become a core stage of modern LLM post-training. PPO (Schulman et al., 2017) estimates per-token advantages using a learned value model, which has to be trained alongside the policy and kept in memory. GRPO (Shao et al., 2024) avoids this by sampling a group of rollouts per prompt and using the group-mean reward as baseline, while RLOO (Kool et al., 2019; Ahmadian et al., 2024) uses the leave-one-out mean of the remaining rollouts instead. Subsequent work identifies biases from length and reward normalization and modifies the sampling distribution, surrogate objective, or optimization procedure for greater stability (Liu et al., 2025; Yu et al., 2025; Xie et al., 2025b; Tajwar et al., 2026). These methods still use a prompt-level baseline and assign the same sequence-level advantage to every token of a rollout, resulting in a coarse learning signal. As we show in Section 4, this baseline captures prompt

difficulty well, but it is shared across the rollouts of a group and does not change as reasoning progresses. CIPS retains the leave-one-out group structure, but replaces the constant baseline with a prefix-dependent score that measures the likelihood of correct answers relative to incorrect ones.

Process reward models. To provide a finer-grained learning signal, process reward models (PRMs) evaluate individual reasoning steps instead of only the final outcome. Early PRMs were trained on human step-level annotations (Uesato et al., 2022; Lightman et al., 2024), and later work scaled this up with labels derived automatically from Monte Carlo (MC) continuations (Wang et al., 2024). Human annotations are expensive to acquire, whereas synthetic labels depend on the policy that generated them and can become stale as the trained policy improves (Yuan et al., 2025). PRIME (Cui et al., 2026) addresses staleness by updating an implicit PRM (Yuan et al., 2025) online from outcome labels, at the cost of training and serving an additional model. Conceptually related, Lee et al. (2026) propose CPMI, which measures how a reasoning prefix changes answer discrimination relative to the prompt alone, and use it as supervision to train a PRM. CIPS instead uses within-group answer likelihoods directly as a baseline during policy optimization, and thus requires neither an additional reward model nor offline label generation.

Sampled continuations. Instead of relying on a learned proxy, MC methods estimate the value of intermediate states online by sampling continuations from the current policy (Kazemnejad et al., 2025; Bachmann et al., 2026). This yields on-policy value estimates, but sampling at many steps along a sequence multiplies rollout cost (Yuan et al., 2025). Tree-based methods reduce this cost by branching at intermediate steps and sharing prefixes between rollouts, deriving step-level advantages from the resulting subtrees (Ding & Ye, 2026), especially for multi-turn agentic tasks (Ji et al., 2026; Feng et al., 2026b). These approaches, however, depart from independent group sampling, which requires custom rollout scheduling and introduces dependencies between rollouts. CIPS keeps independent sampling and therefore integrates into existing, highly optimized synchronous or asynchronous training setups without changes to rollout generation.

Likelihood-based credit assignment. A cost-efficient alternative scores each reasoning prefix by the likelihood of the known correct answer, without sampling any continuations (Zhou et al., 2026; Kwiatkowski et al., 2026; Tao et al., 2026; Xie et al., 2026). TIPS (Xie et al., 2026) uses changes in a teacher model's answer likelihood as a potential-based turn reward (Ng et al., 1999). All of these methods rely on the correct answer's likelihood only, which also varies with context and answer surface form (Zhao et al., 2021; Kuhn et al., 2023). Outside of RL, such effects are typically corrected by normalizing against a content-free or domain-level input (Zhao et al., 2021; Holtzman et al., 2021). CIPS instead uses plausible incorrect answers from the same group as a control, and enters the objective as a baseline rather than as a reward, leaving the optimized objective unchanged.

7 Conclusion

This work introduced the correct-incorrect prefix score (CIPS), a prefix-dependent baseline for reinforcement learning with verifiable rewards. CIPS contrasts the likelihood of correct answers with that of plausible incorrect answers from the same rollout group, requiring neither a learned reward model nor additional rollouts. Compared to single-answer likelihoods, CIPS is more robust to spurious prefix content, such as answer readiness and output format, retains the prompt-difficulty information of the group mean reward, and separates successful from failed rollouts early in the reasoning trace. On mathematical and combinatorial reasoning tasks, CIPS consistently improves over outcome-only Dr. GRPO and prior dense-credit methods, and reaches Dr. GRPO's final performance with about 25% fewer training steps, at almost no added step time and memory.

Limitations. CIPS relies on tasks with short, verifiable final answers that can be scored as a continuation of a reasoning prefix. Extending it to open-ended outputs, such as proofs or code, would require a different notion of answer banks. Furthermore, the admissibility of CIPS holds only under idealized on-policy sampling. In practice, with clipping and dynamic sampling, part of the improvement stems from changes in the scale and shift of the advantage distribution, while the remaining gain is attributable to prefix-dependent credit assignment. Finally, our experiments are limited to models of up to 4B parameters, and it remains to be shown how the gains of CIPS scale to larger models, longer training, and multi-turn agentic tasks.

Reproducibility statement

Section 3 fully specifies CIPS, including the answer-bank construction, the score itself, and the online fitting of the baseline coefficient. Appendix A states the assumptions and gives the complete proof that CIPS is an admissible baseline. All models (Qwen3-1.7B-Base and Qwen3-4B-Base) and datasets (DAPO-Math-17k, MATH-500, Minerva Math, AIME 2024 and 2025, and Countdown) are publicly available. Appendix C reports the hyperparameters, compute setup, and training and evaluation protocol for all compared methods. For the static analysis in Section 4, Appendix B lists the inserted sentences and describes the continued-sampling check. All main results are averaged over three training seeds, and Appendix C.1 reports per-seed results. Since CIPS only replaces the baseline in a standard Dr. GRPO pipeline, it can be implemented directly from these descriptions. Additionally, we provide a reference implementation publicly available at `https://anonymous.4open.science/r/cips-65EB`.

AI use statement

In this work, we used generative AI tools to provide feedback on experiment design, assist in the writing of proofs, and implement experiments. We have not used generative AI tools for formulating or providing critical ingredients for mathematical claims, proposing hypotheses, cleaning and reformatting datasets, interpret results and generating synthetic data sets and translation are not applicable to this work. Additionally, we used generative AI tools for grammatical checks and to improve reading flow on sentence structure. We have reviewed all AI-assisted work. LLM-generated code was reviewed and tested for correctness with software tests. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024.

Gregor Bachmann, Yichen Jiang, Seyed-Mohsen Moosavi-Dezfooli, and Moin Nabi. The potential of cot for reasoning: A closer look at trace dynamics. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, 2026.

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.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Zheng Ding and Weirui Ye. Treegrpo: tree-advantage grpo for online rl post-training of diffusion models. In *International Conference on Learning Representations*, volume 2026, pp. 10877–10889, 2026.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, 2026a.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026b.

Kanishk Gandhi, Denise H J Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah Goodman. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 23–29 Jul 2023.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5, 2004.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 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. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, November 2021.

Prime Intellect. prime-rl, 2025.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 87362–87388, 2026.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025.

Wouter Kool, Herke Van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In *ICLR 2019 Workshop on Deep Reinforcement Learning Meets Structured Prediction*, 2019.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023.

Ariel Kwiatkowski, Natasha Butt, Ismail Labiad, Julia Kempe, and Yann Ollivier. Likelihood-based reward designs for general llm reasoning. *arXiv preprint arXiv:2602.03979*, 2026.

Nakyung Lee, Sangwoo Hong, and Jungwoo Lee. Efficient process reward modeling via contrastive mutual information. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2026.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

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. In *Conference on Language Modeling (COLM)*, 2025.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. Trace: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Siddarth Venkatraman, Matthieu Dinot, and Laurence Aitchison. Le critique: Privileged value functions for llm reinforcement learning. *arXiv preprint arXiv:2608.16739*, 2026.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, August 2024.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. In *International Conference on Learning Representations*, volume 2026, 2026.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. Capo: Towards enhancing llm reasoning through generative credit assignment. *arXiv preprint arXiv:2508.02298*, 2025a.

Yutao Xie, Nathaniel Thomas, Nick Hansen, Yang Fu, Li Li, and Xiaolong Wang. Tips: Turn-level information-potential reward shaping for search-augmented llms. In *International Conference on Learning Representations*, volume 2026, pp. 156549–156584, 2026.

Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. In *Forty-second International Conference on Machine Learning*, 2025b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 13–19 Jul 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 18–24 Jul 2021.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. In *International Conference on Learning Representations*, volume 2026, pp. 69689–69712, 2026.

A PROOF OF ADMISSIBILITY FOR CIPS AS BASELINE

Fix a prompt x and target rollout y_i, and assume that the K rollouts in its group are conditionally independent samples from $\pi_\theta(\cdot \mid x)$. Assume $\mathcal{A}_i^-$ is nonempty, so CIPS is used, and consider a token u in the segment following boundary k. Let $\mathcal{H}_{i,u}$ contain x, the target history $y_{i,<u}$, the complete peer rollouts, the reference answer, scorer parameters, and all statistics and coefficients computed before the group. The baseline $b_{i,u} = c\,\Phi_i(k)$ is $\mathcal{H}_{i,u}$-measurable: its only target-dependent input is the earlier prefix $s_{i,k}$. Its remaining inputs are the fixed reference answer and independent peer rollouts.

Conditional independence therefore leaves the target token distributed as $y_{i,u} \sim \pi_\theta(\cdot \mid x, y_{i,<u})$. Hence

$$\mathbb{E}[b_{i,u}\nabla_\theta \log \pi_\theta(y_{i,u} \mid x, y_{i,<u}) \mid \mathcal{H}_{i,u}] \tag{9}$$

$$= b_{i,u} \sum_v \pi_\theta(v \mid x, y_{i,<u}) \nabla_\theta \log \pi_\theta(v \mid x, y_{i,<u}) \tag{10}$$

$$= b_{i,u} \nabla_\theta \sum_v \pi_\theta(v \mid x, y_{i,<u}) = 0. \tag{11}$$

Taking expectations and summing over target tokens shows

$$\mathbb{E}\left[\sum_u \big(R_i - b_{i,u}\big) \nabla_\theta \log \pi_\theta(y_{i,u} \mid x, y_{i,<u})\right] \tag{12}$$

$$= \mathbb{E}\left[\sum_u R_i \nabla_\theta \log \pi_\theta(y_{i,u} \mid x, y_{i,<u})\right] = \nabla_\theta \mathbb{E}_{y_i \sim \pi_\theta(\cdot|x)}[R(x, y_i)]. \tag{13}$$

Thus, CIPS does not change the expected on-policy REINFORCE gradient, and averaging over $x \sim \mathcal{D}$ recovers $\nabla_\theta \mathcal{J}(\theta)$. The same argument holds for the adaptive coefficient because group g uses the lagged c_{g-1}. It also permits a current-policy scorer, provided its outputs are detached and its parameters remain fixed while the sampled group is processed.

This proposition concerns the unclipped, unfiltered on-policy estimator. The empty-bank fallback uses the Dr. GRPO group-mean advantage and therefore inherits that estimator instead. When the group-mean advantage $R_i - K^{-1}\sum_j R_j$ is used for every target, its expected score-function gradient is $(K-1)/K$ times the outcome-only gradient under independent group sampling. This constant scaling originates from GRPO and does not change the ideal optimum.

B ROBUSTNESS TO SPURIOUS CONTEXT CONTENT

We select ten short sentences that signal answer readiness, remind the model of the requested format, express confidence or uncertainty, or prompt further checking or reasoning, without supplying an answer or derivation. At 3,072 saved reasoning prefixes from 1,024 prompts, we measure correct- and incorrect-answer bank log likelihoods before and after each insertion, alongside their difference (CIPS). We score two targets: *answer only* measures answer content after a supplied `\boxed{` opening, while *full boxed* measures the complete `\boxed{answer}` continuation. We test whether the insertion of these cues changes the outcome of prefixes. We see a largely unchanged outcome (87.6% of cases), with a flip to an unsuccessful rollout being more common. We attribute this to the confusing nature of these sentences when inserted in a reasoning chain. The full results of the robustness analysis of Section 4 are presented in Table 3.

Full-target vs. within-template scoring. Whether a correct-only method includes the opening answer template in the likelihood scoring or not changes what is measured. Full-target scoring, as in Xie et al. (2026), essentially reduces the answer likelihood to a readiness signal. It barely correlates with rollout outcome ($\rho = 0.037$ for correct and $\rho = 0.089$ for incorrect likelihood), and is highly sensitive to spurious prefix content. Measuring the likelihood of the within-template answer reduces the impact of spurious content and increases correlation with rollout outcome. For CIPS, the influence of the template conditioning cancels between banks, so we recommend the use of full-box scoring, since it is easier to adapt to new answer templates.

Table 3: Inserted cues and their effect on signals to test signal robustness. RMS of score with and without the cue is measured and normalized by the typical RMS of the signal along rollouts.

Insertion	Method	Full box scoring?	RMS	Natural RMS	Ratio
Ready *"Now we can answer."*	Correct only	No	1.846	2.257	0.818
	Incorrect only	No	1.441	1.913	0.753
	Correct only	Yes	14.101	7.153	1.971
	Incorrect only	Yes	13.726	6.889	1.992
	CIPS	Yes	1.384	2.507	**0.552**
Format *"We need to provide the answer in \boxed{} format."*	Correct only	No	1.705	2.257	0.756
	Incorrect only	No	1.337	1.913	0.699
	Correct only	Yes	14.369	7.153	2.009
	Incorrect only	Yes	14.050	6.889	2.039
	CIPS	Yes	1.377	2.507	**0.549**
Neutral *"This is a mathematical problem."*	Correct only	No	1.827	2.257	0.809
	Incorrect only	No	1.428	1.913	0.747
	Correct only	Yes	12.854	7.153	1.797
	Incorrect only	Yes	12.501	6.889	1.815
	CIPS	Yes	1.366	2.507	**0.545**
Confident *"We have completed the reasoning and verified the result."*	Correct only	No	1.898	2.257	0.841
	Incorrect only	No	1.453	1.913	0.760
	Correct only	Yes	14.697	7.153	2.055
	Incorrect only	Yes	14.305	6.889	2.076
	CIPS	Yes	1.462	2.507	**0.583**
Answer follows *"The final answer follows."*	Correct only	No	1.867	2.257	0.827
	Incorrect only	No	1.445	1.913	0.755
	Correct only	Yes	14.426	7.153	2.017
	Incorrect only	Yes	14.044	6.889	2.039
	CIPS	Yes	1.415	2.507	**0.564**
Plain format *"We need to provide the answer in the requested format."*	Correct only	No	1.847	2.257	0.818
	Incorrect only	No	1.428	1.913	0.747
	Correct only	Yes	14.436	7.153	2.018
	Incorrect only	Yes	13.998	6.889	2.032
	CIPS	Yes	1.467	2.507	**0.585**
Not ready *"We need to do a couple more validations before we can answer."*	Correct only	No	1.894	2.257	0.839
	Incorrect only	No	1.462	1.913	0.764
	Correct only	Yes	11.540	7.153	1.613
	Incorrect only	Yes	11.113	6.889	1.613
	CIPS	Yes	1.473	2.507	**0.588**
Uncertain *"We are unsure whether our reasoning is correct."*	Correct only	No	1.875	2.257	0.831
	Incorrect only	No	1.468	1.913	0.767
	Correct only	Yes	11.985	7.153	1.675
	Incorrect only	Yes	11.603	6.889	1.684
	CIPS	Yes	1.453	2.507	**0.580**
Check arithmetic *"We need to check the arithmetic carefully."*	Correct only	No	1.895	2.257	0.840
	Incorrect only	No	1.461	1.913	0.764
	Correct only	Yes	11.350	7.153	1.587
	Incorrect only	Yes	10.916	6.889	1.585
	CIPS	Yes	1.449	2.507	**0.578**
Continue *"We need to continue reasoning about this problem."*	Correct only	No	1.962	2.257	0.869
	Incorrect only	No	1.535	1.913	0.802
	Correct only	Yes	12.218	7.153	1.708
	Incorrect only	Yes	11.822	6.889	1.716
	CIPS	Yes	1.512	2.507	**0.603**

C ADDITIONAL EXPERIMENTS AND RESULTS

The experiment runs share the identical optimization, rollout, reward, and evaluation configurations, as detailed in Table 4. We use the `prime-rl` framework from PrimeIntellect as a base for implementation (Intellect, 2025).

Table 4: Hyperparameters shared across experimental runs.

Hyperparameter	**Value**
Base model	`Qwen/Qwen3-1.7B-Base`
Training dataset	`open-r1/DAPO-Math-17k-Processed` (English train split)
Training steps	240
Rollout batch size	128 accepted rollouts per update
Rollouts per problem	8
Maximum completion length	2,048 tokens
Rollout temperature	1.0
Top-p	1.0
Optimizer	AdamW
Learning rate	1×10^{-6}
LR schedule	Constant, without warm-up
Adam coefficients	$(\beta_1 = 0.9,\ \beta_2 = 0.999)$
PPO ratio clipping	$[0.8, 1.2]$, i.e., $(\epsilon_{\text{low}} = \epsilon_{\text{high}} = 0.2)$
TIPS: α	0.1

C.1 EVALUATION PROTOCOL AND STATISTICAL ANALYSIS

Benchmarks. We evaluate on MATH500 (500 problems), Minerva Math (272), AIME (60; combined from 2024 and 2025). Countdown is evaluated on 500 held-out problems with four (Qwen3-1.7B) or five (Qwen3-4B) input numbers. All methods use the final checkpoint from each run and identical evaluation settings: 16 samples per problem (8 for MATH500), temperature 1.0, a maximum of 2,048 generated tokens. Answers are extracted from the last `\boxed{}` and checked with symbolic equivalence via `math-verify`. Countdown answers are checked by evaluating the expression and verifying that each input number is used at most once.

Metrics. For a problem with n samples of which c are correct, we use the unbiased estimator of Chen et al. (2021),

$$\text{pass@}k = 1 - \binom{n-c}{k} \Big/ \binom{n}{k}, \qquad k \le n, \tag{14}$$

with $k \in \{1, 4\}$ for MATH500 and $k \in \{1, 8\}$ otherwise. A benchmark score for one training run is the mean over its problems. Avg is the unweighted mean over the columns of Table 1 and Table 6.

Standard errors. Table entries report the mean over seeds $\pm$ the standard error $\hat{\sigma}/\sqrt{S}$, where $\hat{\sigma}$ is the sample standard deviation over the $S = 3$ seeds. In Table 5 we show the per-seed averages for the 120 step runs. It shows that every CIPS seed wins over all seeds from other methods, showing a reproducible performance gain.

C.2 MATH RESULTS WITH SHORTER TRAINING HORIZON

Besides the performance reported in Table 1, which reports the performance of CIPS with those of various other methods after 240 steps, we also evaluated all methods for a shorter training horizon with 120 steps, to see if the methods separate early or late (Table 6). For Qwen3-1.7B, we see that CIPS clearly outperforms all other methods already after 120 steps, leading on all six metrics and improving by 1.8 points over Dr. GRPO on average. For the shorter training we see the same ordering as for the 240-step training runs, with CIPS being the strongest method, followed by TIPS and Dr. GRPO, and VinePPO already trailing behind.

Table 5: Per-seed aggregate scores for Qwen3-1.7B. Each CIPS seed performs better than any seed of another method.

	Average Performance		
Method	**Seed 0**	**Seed 1**	**Seed 2**
Qwen3-1.7B	23.12	22.99	22.99
Dr. GRPO	34.21	34.47	33.86
TIPS	35.14	35.72	34.83
TIPS (answer-only)	35.79	34.40	34.96
TIPS (contrastive)	33.73	34.75	34.60
CIPS	36.01	35.84	36.19

Table 6: Evaluation results (pass@1 and pass@k in %; pass@4 for MATH-500 and pass@8 for remaining benchmarks) across Qwen3-1.7B and Qwen3-4B after **120 steps**. For VinePPO we use a 4x time budget. Values are means over three training seeds ± standard error. **Bold** indicates the highest mean, underline indicates the second-highest. Rows marked with Δ_{baseline} display point differences relative to Dr. GRPO.

	MATH-500		AIME		Minerva		
Method	**p@1**	**p@4**	**p@1**	**p@8**	**p@1**	**p@8**	**Avg**
Qwen3-1.7B	31.0 ±0.6	60.9 ±0.2	0.8 ±0.0	4.4 ±0.3	9.5 ±0.1	31.5 ±0.1	23.0 ±0.0
Dr. GRPO	54.8 ±0.3	76.2 ±0.4	3.1 ±0.2	10.3 ±0.7	19.2 ±0.3	41.5 ±0.3	34.2 ±0.2
VinePPO	37.5 ±0.4	68.3 ±0.6	1.3 ±0.2	7.1 ±0.4	12.3 ±0.2	34.3 ±0.2	26.8 ±0.3
TIPS	56.8 ±0.2	76.1 ±0.3	3.4 ±0.1	12.7 ±0.4	20.5 ±0.2	41.8 ±0.6	35.2 ±0.3
CIPS	**57.4** ±1.0	**77.4** ±0.6	**3.5** ±0.4	**13.8** ±2.4	**21.2** ±0.5	**42.9** ±0.7	**36.0** ±0.1
Δ_{baseline}	+2.6	+1.2	+0.4	+3.5	+2.0	+1.4	+1.8
Qwen3-4B	38.5 ±0.3	74.1 ±0.2	4.1 ±0.2	17.8 ±0.1	18.0 ±0.2	46.8 ±0.9	33.2 ±0.2
Dr. GRPO	70.9 ±0.3	86.9 ±0.3	9.7 ±0.2	24.5 ±1.3	31.5 ±0.3	53.1 ±0.7	46.1 ±0.4
VinePPO	54.0 ±0.4	82.0 ±0.8	6.7 ±0.4	21.9 ±0.5	23.5 ±0.3	50.5 ±0.4	39.8 ±0.5
TIPS	72.2 ±0.1	86.8 ±0.1	**10.0** ±0.2	24.5 ±0.3	31.7 ±0.2	53.1 ±0.1	46.4 ±0.1
CIPS	**72.4** ±0.1	**87.0** ±0.4	9.8 ±0.9	**26.6** ±0.1	**33.3** ±0.1	**53.7** ±0.2	**47.2** ±0.1
Δ_{baseline}	+1.5	+0.1	+0.1	+2.1	+1.8	+0.6	+1.1

Similarly, CIPS is the strongest method on five of six metrics for the Qwen3-4B training, being only close second to TIPS on AIME pass@1. On average, CIPS again is the strongest method with a lead of 1.1 points over Dr. GRPO.

C.3 Countdown evaluation

Qwen3-1.7B is trained and evaluated on four-number problems, whereas Qwen3-4B is trained and evaluated on five-number problems of greater difficulty.

Table 7 reports pass@1, pass@4, and pass@8 on Countdown, averaged over three seeds. The results follow the same pattern as for mathematical reasoning. For Qwen3-1.7B, CIPS achieves the best score on every metric, raising the average from 29.0 to 29.8 (+0.8 points, +2.8%). The gain over Dr. GRPO grows with k, from +0.2 at pass@1 to +1.6 at pass@8, showing a better balance between exploitation and exploration. Outcome-only training tends to sharpen the policy around a few high-reward solutions, which improves pass@1 but at the cost of diversity of sampled answers (Yue et al., 2025). TIPS matches CIPS on pass@1 but falls behind for larger k.

On the harder five-number problems with Qwen3-4B, the improvement of CIPS is larger and more uniform. CIPS improves over Dr. GRPO on every metric by 1.1 to 1.4 points, lifting the average from 28.9 to 30.1 (+1.2 points, +4.2%). TIPS, in contrast, falls slightly below Dr. GRPO on all

Table 7: Evaluation results on the Countdown benchmark across Qwen3-1.7B and Qwen3-4B, including accuracy (pass@1, pass@4, pass@8, and Avg in %). Values are means over three training seeds ± standard error. **Bold** indicates best performance; underline indicates second-best within each model scale. Row marked with Δ_{baseline} display point differences relative to baseline.

	Qwen3-1.7B Diff: 4				**Qwen3-4B** Diff: 5			
Method	**p@1**	**p@4**	**p@8**	**Avg**	**p@1**	**p@4**	**p@8**	**Avg**
Base model	0.7 ±0.0	2.7 ±0.1	5.1 ±0.3	2.8 ±0.1	0.4 ±0.0	1.4 ±0.0	2.7 ±0.1	1.5 ±0.0
Dr. GRPO	26.1 ±0.1	30.0 ±0.0	30.7 ±0.1	29.0 ±0.0	18.3 ±0.3	31.3 ±0.6	36.9 ±1.5	28.9 ±0.6
TIPS	**26.3** ±0.3	30.3 ±0.5	31.3 ±0.8	29.3 ±0.5	18.2 ±0.9	30.9 ±0.9	36.3 ±1.0	28.5 ±0.9
CIPS	**26.3** ±0.1	**30.8** ±0.2	**32.3** ±0.2	**29.8** ±0.2	**19.4** ±0.8	**32.6** ±1.1	**38.3** ±1.1	**30.1** ±1.0
Δ_{baseline}	+0.2	+0.8	+1.6	+0.8	+1.1	+1.3	+1.4	+1.2

metrics at this scale. Across both tasks and model sizes, CIPS is the only method that consistently outperforms outcome-only training.

C.4 Extended ablation

Table 8 reports additional ablations for Qwen3-1.7B, each trained for 120 steps and averaged over three seeds.

Affine baseline. Instead of scaling CIPS with an online-fitted coefficient, we use the affine baseline $\beta_0 + \beta_R \bar{R}_{-i} + \beta_\Phi \Phi_i(k)$, whose coefficients are fitted offline by least-squares regression onto the rollout reward R. Despite this fit to R, the affine baseline performs worse than Dr. GRPO on three of four metrics, reaching only 52.7 pass@1 on MATH-500 compared with 54.8 (Table 8a). A good offline fit to the reward, thus, does not by itself translate into better updates during training. This suggests that the benefits of both are not compatible with a static combination.

Fixed coefficient. To test whether adapting the coefficient during training matters, we fix $c = 0.045$, the mean of the online-fitted coefficient over the CIPS runs (Table 8a). The fixed coefficient performs on par with the online fit, scoring higher on both AIME metrics and lower on MATH-500 pass@4, so we observe no consistent benefit from adapting c during training. The online fit nonetheless removes the need to tune c for each model and dataset, since the value of 0.045 is itself only known from the adaptive runs.

Advantage scale and shift. Since $c\,\Phi_i(k)$ is small, with 10th and 90th percentiles of -0.4 and 0.4, CIPS shifts and rescales the advantage distribution relative to Dr. GRPO: correct rollouts receive larger positive advantages, and incorrect rollouts receive advantages close to zero. To separate this effect from prefix-dependent credit assignment, we train Dr. GRPO with advantages transformed to match those of CIPS, once with a constant offset and scale (static) and once with an online rescaling that tracks the CIPS advantage statistics during training (dynamic). Both variants improve over Dr. GRPO on all four metrics but remain below CIPS on all metrics except AIME pass@8, where the dynamic variant reaches 14.3 compared with 13.8 (Table 8a). On MATH-500, they reach 56.1 and 56.7 pass@1, compared with 57.4 for CIPS. Matching the advantage distribution thus explains part of the improvement, but not all of it.

CIPS at Dr. GRPO scale. Conversely, we rescale and offset CIPS to match advantages of Dr. GRPO. For each group, we compute both the Dr. GRPO and the CIPS advantages and rescale and shift the CIPS advantages to match those of the Dr. GRPO advantages (Table 8b). This variant falls between the two methods: it improves over Dr. GRPO on all four metrics, for example, from 54.8 to 55.7 pass@1 on MATH-500, but remains below CIPS on all four. Together with the rescaled Dr. GRPO runs above, this indicates that both components contribute: the advantage scale accounts for part of the gain, and the prefix-dependent baseline improves over Dr. GRPO even when the advantage magnitude is matched.

Table 8: Ablations on MATH-500 and AIME. Each sub-block groups relevant pairwise comparisons against reference configurations. Every configuration ran for three seeds.

	MATH-500		AIME	
Configuration	**p@1**	**p@4**	**p@1**	**p@8**
a) Default Comparisons				
Affine combination $\beta_0 + \beta_R\, R + \beta_\Phi\, \Phi$	52.7	75.1	2.8	11.7
Dr. GRPO (CIPS matched, static)	56.1	77.0	3.2	12.4
Dr. GRPO (CIPS matched, dynamic)	56.4	76.9	3.4	**14.0**
CIPS (fixed $c = 0.045$)	57.3	76.3	**4.1**	13.9
CIPS	**57.4**	**77.4**	3.5	13.8
b) Baseline Comparisons				
Dr. GRPO	54.8	76.2	3.1	10.3
CIPS (Dr. GRPO matched, dynamic)	**55.7**	**76.6**	**3.4**	**12.9**
c) TIPS Variants				
Default	**56.6**	76.2	**3.6**	12.8
Contrastive	55.2	**76.5**	2.8	11.4
Within-template scoring	55.9	76.2	3.2	**13.8**

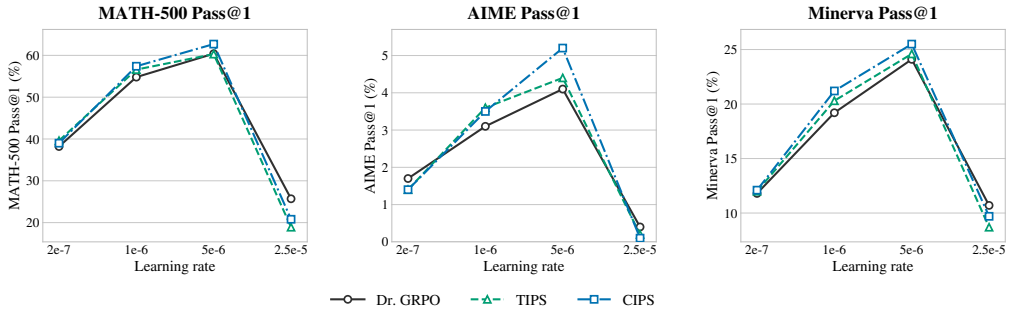


Figure 4: Pass@1 performance across learning rates on MATH-500, AIME, and Minerva. CIPS achieves the highest score on all three benchmarks, even when each method gets the optimal learning rate.

TIPS variants. Finally, we compare three answer potentials for TIPS: the default full-target scoring, as implemented in the authors' published code; within-template scoring (Section 2); and a contrastive variant that replaces the correct-only likelihood with CIPS (Equation 5). The default performs best on both pass@1 metrics, while within-template scoring leads on AIME pass@8 and the contrastive variant on MATH-500 pass@4 (Table 8c). We therefore use the default in the main comparison. The contrastive variant performs worst on three of four metrics, suggesting that the CIPS contrast is more effective as a baseline than as a shaping potential.

Learning rate sensitivity. Across 2×10^{-7}, 1×10^{-6}, and 5×10^{-6}, CIPS has the highest mean over the four metrics in Figure 4. This shows that CIPS provides a robust improvement over other methods, instead of simply scaling their advantages.

C.5 Memory overhead

The scorer of TIPS and CIPS adds very little memory overhead. In our setup with three or four A100 GPUs, the scorer runs as a vLLM server co-hosted on the trainer's GPU, and its KV cache consistently stays below 5 GB. Because training is asynchronous, each group is scored in parallel with the generation of subsequent rollouts.