REINFORCEMENT LEARNING WITH VERIFIABLE YET NOISY REWARDS UNDER IMPERFECT VERIFIERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) replaces costly human labeling with automated verifiers. To reduce verifier hacking, many RLVR systems binarize rewards to $\{0, 1\}$, but imperfect verifiers inevitably introduce *false negatives* (FN, rejecting correct answers) and *false positives* (FP, accepting incorrect ones). We formalize verifier unreliability as a stochastic reward channel with asymmetric noise rates ρ_0 and ρ_1—the FP rate and the FN rate, respectively. From this abstraction we derive two lightweight corrections: (i) a *backward* correction that yields an unbiased surrogate reward and thus an unbiased policy-gradient estimator in expectation, and (ii) a *forward* correction that reweights score-function terms so the expected update aligns with the clean gradient direction and requires only the FN rate. We implement both as lightweight hooks in a group relative policy optimization pipeline, both corrections improve RLVR reasoning under synthetic and real verifier noise, with the forward variant being more stable under heavier noise. We further use a lightweight appeals mechanism to estimate FN rates online and show that the correction can complement stronger verifier-side baselines.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) offers a scalable paradigm for improving the reasoning abilities of Large Language Models (LLMs) by replacing expensive human annotation with automated feedback (Wen et al., 2025). In this problem, a policy is trained using rewards from a verifier that automatically checks the correctness of a model's output (Shao et al., 2024). The efficacy of this approach, however, heavily depends on the verifier's reliability. Emerging evidence reveals that verifiers are systematically fallible in two critical ways: they can accept incorrect solutions (*false positives; FPs*) or reject correct ones (*false negatives; FNs*) (Xu et al., 2025; Zhao et al., 2025).

FPs have been widely documented as a vulnerability of LLM-based verifiers. Recent studies showed that LLM judges can be swayed by superficial cues—e.g., popular specialized verifiers, such as GPT-4o, give *35 % – 66.8%* FP rate when the answer starts with *Let's solve this problem step by step* (Zhao et al., 2025; Shi et al., 2025; 2024; Chen et al., 2024). Conversely, *FNs* are common with rule-based verifiers. These checkers, while highly precise, are often brittle; they may reject valid solutions that are formatted differently, expressed in an algebraically equivalent form, or embedded in explanatory text (Hugging Face, 2025). A recent analysis of a math-RL dataset found that over *38%* of responses flagged as incorrect by a rule-based system were in fact correct, a gap that a lightweight LLM verifier could partially close (Xu et al., 2025). Both FPs and FNs materially degrade RLVR training: FNs deprive the agent of informative gradients and slow convergence, while FPs reward hackable patterns and inflate returns during policy optimization (Xu et al., 2025; Huang et al., 2025).

Motivated by these failures, we address these challenges by explicitly *treating verifier errors as noise* in the reward signal. We model the verifier as a stochastic reward channel that corrupts the true, latent reward with verifier-conditional probabilities—the FP rate and the FN rate. Building on this formulation, we derive two estimators to counteract this noise. Our first method, *noisy Policy Gradient with Backward Correction (PGBC)*, inverts the noise process to compute an unbiased estimator of the true reward, which can be used as a drop-in replacement in any advantage-estimation in RL. Our second method, *noisy Policy Gradient with Forward Correction (PGFC)*, directly reweights the terms of the policy gradient to ensure its expected direction aligns with the clean gradient. This approach

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.

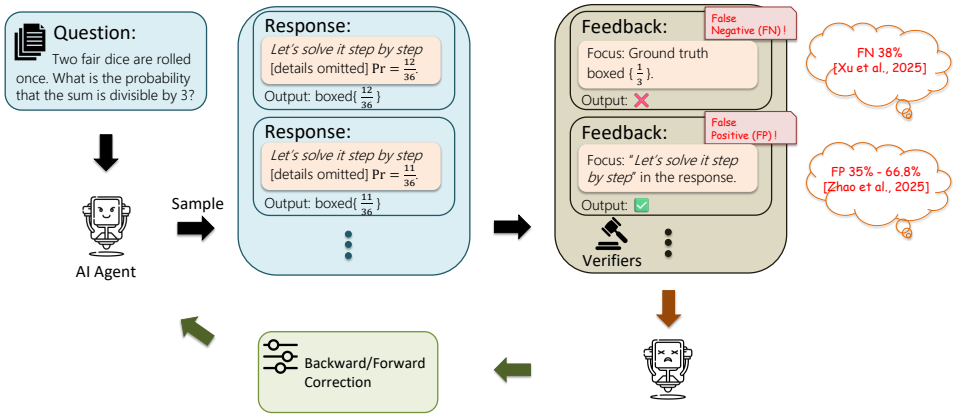


Figure 1: Verifier-noise flow in RLVR. An AI agent produces candidate solutions that are scored by automated verifiers. While verifiers would yield *false negatives* ($\frac{12}{36}$ vs. $\frac{1}{3}$, reaching 38% rates (Xu et al., 2025)) and *false positives* (mislead by "Let's solve it step by step...", reaching $35\% - 66.8\%$ rates (Zhao et al., 2025)), confusing the agent; applying our backward/forward corrections restores correct signals.

requires only an estimate of the FN rate to achieve directional correctness, making it suitable for the often harder-to-estimate FP rate (Xu et al., 2025). The pipeline is illustrated in Figure 1.

We integrate these corrections into a group relative policy optimization (GRPO) (Shao et al., 2024)-based RLVR pipeline and demonstrate their effectiveness on competitive math reasoning benchmarks (Shao et al., 2024). Both methods consistently outperform uncorrected training and achieve performance nearly equivalent to that of noise-free scenarios, with the forward correction offering the fastest and most stable convergence. We also propose a practical online estimation scheme for the FN rate: a querying mechanism where a low-cost LLM verifier like TinyV (Xu et al., 2025) re-evaluates outputs rejected by a primary rule-based checker. This hybrid approach provides a reliable estimate of the FN rate with minimal computational overhead, recovering near-oracle performance. Furthermore, we stress-test our approach under complex, non-i.i.d. noise distributions that violate our FP and FN modeling, but both PGBC and PGFC corrections remain robust and effective even in these more challenging conditions. Our contributions are: (i) a formal verifier-channel model for RLVR that captures the prevalent FP and FN errors; (ii) backward and forward corrections derived from RL principles to calibrate policy learning; and (iii) a practical implementation for online noise estimation that makes these corrections viable in real-world systems.

2 Related Works

Reasoning with LLMs. A large body of work improves LLM reasoning through prompting and search at inference time. Chain-of-Thought (CoT) prompting elicits step-by-step rationales and boosts arithmetic and commonsense reasoning (Wei et al., 2022), while self-consistency aggregates diverse reasoning paths to further improve robustness (Wang et al., 2023). Decomposition-based prompting, such as Least-to-Most, solves complex problems via ordered sub-problems (Zhou et al., 2023); search-based schemes like Tree-of-Thoughts explicitly explore and evaluate alternative reasoning branches (Yao et al., 2023). Orthogonally, training-time verifier signals can be used to re-rank candidate solutions, as in Cobbe et al. (2021). Our work is complementary: rather than proposing a new inference-time strategy or re-ranking scheme, we address *how* to perform *policy optimization* when the reward itself—supplied by a verifier—is noisy.

RLVR and verifier reliability. Recent math-RL pipelines combine on-policy RL (e.g., GRPO) with programmatic or LLM-based verifiers to yield verifiable rewards (Mroueh, 2025). However, LLM-as-a-judge is prone to systematic biases, including position bias and prompt-based attacks, producing *FPs* (accepting incorrect solutions) (Gu et al., 2024; Thakur et al., 2024; Shi et al., 2025; Zhao et al., 2025; Shi et al., 2024). Conversely, exact-match or brittle parsers in rule-based checkers can miss algebraic equivalences and text-formatted answers, leading to widespread *FNs* (Li et al., 2025; Huang et al., 2025). A recent study showed that over 38% of model responses labeled incorrect

by a rule-based pipeline were in fact correct, and introduced a lightweight verifier (TinyV) that recovers many such cases with minimal overhead (Xu et al., 2025). Prior efforts therefore improved the *verifier* (e.g., better judges or appeals) or evaluation suites; in contrast, we treat verifier errors as verifier-conditional noise and directly *correct the policy gradient* so that learning remains aligned with the clean objective even when the verifier is imperfect.

Learning with noisy labels. While our method is derived from RL, it shares intuition with the literature on learning with label noise. Wang et al. (2020) modeled reward corruption via a class-conditional confusion matrix and derived an unbiased surrogate reward for Q-learning algorithms to solve control tasks. In supervised learning, a central thread is to make empirical-risk minimization robust either by *correcting the loss* given a noise-transition model or by *avoiding* or *down-weighting* suspected noisy examples (Song et al., 2022; Li et al., 2021). The former includes the unbiased-risk estimators of Natarajan et al. (2013) and the now-standard loss corrections of Patrini et al. (2017). A second family avoids the noise explicitly by relying on the "small-loss first" memorization dynamics of deep nets: mentor methods (MentorNet) learn a weighting network that feeds cleaner samples to the student, and *Co-teaching* (Han et al., 2018) trains two peers that exchange their selected small-loss examples; semi-supervised hybrids such as *DivideMix* (Li et al., 2020) split data into clean/noisy partitions via mixture modeling and co-train with consistency regularization (Jiang et al., 2018). Unlike them, we target the *policy gradient estimator* in RLVR: we instantiate both backward and forward corrections inside the policy gradient estimator to stabilize and align RL under verifier noise, instead of a direct supervised-loss substitution.

3 Problem Setup and Algorithms

In this section, we introduce the problem formulations and our proposed algorithms.

3.1 Problem Setup

We consider the standard RLVR setting where a stochastic policy $\pi_\theta(y|x)$ generates a response y for a given prompt x, where θ denotes learnable parameters. The goal is to maximize an objective based on the true, or *clean*, reward $R^*(x, y) \in \{0, 1\}$, which indicates whether the response y is genuinely correct. This objective is typically regularized by a KL-divergence term to a reference policy π_{ref} to maintain stability:

$$\max_\theta \; J_\beta(\theta) \;=\; \mathbb{E}_x \, \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\Big[\, R^*(x,y) - \beta \, \mathrm{KL}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x))\,\Big], \tag{1}$$

in which KL denotes the Kullback-Leibler (KL)-divergence. The policy gradient with respect to the unregularized reward is given by the REINFORCE estimator (Sutton et al., 1999), $\nabla_\theta J(\theta) = \mathbb{E}[R^*(x, y) \sum_t G_t]$, where $G_t = \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})$ is the score function for the token at step t, and $y_{<t}$ denotes the sequence of tokens generated prior to step t. In the following, we will denote $R(x, y)$ with R for brevity. In practice, the clean reward R^* is unavailable. Instead, the agent receives a noisy reward $\tilde{R}(x, y) \in \{0, 1\}$ from an automated verifier. We model this verifier as a stochastic *reward channel* that flips the latent clean reward to an observed noisy one. This channel is characterized by asymmetric, verifier-conditional noise rates.

Definition 3.1 (Verifier Reward Channel)**.** The observed verifier reward $\tilde{R}$ is generated from the latent clean reward R^* according to the following conditional probabilities:

$$\mathbb{P}\big(\tilde{R} = 1 \mid R^* = 0\big) = \rho_0, \mathbb{P}\big(\tilde{R} = 0 \mid R^* = 1\big) = \rho_1, \tag{2}$$

where $\rho_0, \rho_1 \in [0, 1)$ and $\rho_0 + \rho_1 < 1$.

The condition $\rho_0 + \rho_1 < 1$ implies that the verifier channel is informative and invertible. Equivalently, the 2×2 binary transition matrix has determinant $1 - \rho_0 - \rho_1 > 0$.

Assumptions and scope of the verifier channel. Definition 3.1 is a deliberately lightweight abstraction of verifier unreliability. It assumes (i) *instance-independent* class-conditional noise rates (ρ_0, ρ_1) that do not vary with (x, y), and (ii) *conditional independence*: given the latent correctness $R^*(x, y)$, the verifier's stochastic flip $\tilde{R}$ is independent of the policy's sampling randomness (hence independent of the score-function terms used by policy gradients). These assumptions are standard

in class-conditional noise models and allow closed-form reward/gradient corrections. In real RLVR deployments, verifier errors can be content-dependent (e.g., spurious phrases can increase FPs), which violates (i) and can partially violate (ii). We therefore view (ρ_0, ρ_1) as *effective* noise rates, and empirically test robustness to mis-specification in Section 4.4; we further estimate ρ_1 online via a low-cost appeals mechanism in Section 4.3. When rates are instance-dependent, the same formulas are treated as correcting an effective marginal channel rather than eliminating all bias; Appendix B gives the resulting covariance residual.

Based on this noise model, we establish a linear relationship between the expected noisy reward and the clean reward, which is presented in Proposition 3.2.

Proposition 3.2 (Connection between Corrupted Rewards and True Rewards)**.** *Under the Verifier Reward Channel model, the expectation of the noisy reward $\tilde{R}$ conditioned on the clean reward R^* is an affine transformation of R^*:*

$$\mathbb{E}[\tilde{R} \mid R^*] \;=\; (1 - \rho_0 - \rho_1)\, R^* + \rho_0.$$

The proof is provided in Appendix D.1. The central challenge of RLVR is that naively optimizing with the noisy reward $\tilde{R}$ leads to a biased policy gradient, causing the policy to learn from mistakes of the verifier. Our goal is to develop policy gradient estimators that use only the observable noisy reward $\tilde{R}$ but the expectation is either identical or parallel to the true policy gradient $\nabla_\theta J(\theta)$. We integrate these estimators within REINFORCE-style policy gradient algorithms widely used for reasoning tasks that compute normalized advantages over a group of sampled trajectories (Shao et al., 2024).

3.2 Noisy Policy Gradient with Backward Correction (PGBC)

Our first proposed approach aims to construct an unbiased estimator of the true reward R^* by "inverting" the noise process. Since the expected noisy reward $\mathbb{E}[\tilde{R} \mid R^*]$ is an affine transformation of the clean reward R^* in Proposition 3.2, we can solve for R^* to derive a corrected reward estimator, $\widehat{R}$, that is unbiased in expectation.

Theorem 3.3 (Oracle-Rate Unbiased Reward Estimator for the Binary Verifier Channel)**.** *Given the verifier channel with known noise rates ρ_0 and ρ_1 with $\rho_0 + \rho_1 < 1$, the estimator*

$$\widehat{R} = \frac{\tilde{R} - \rho_0}{1 - \rho_0 - \rho_1} \tag{3}$$

is an unbiased estimator of the true reward $R^*(x, y)$, i.e., $\mathbb{E}[\widehat{R}] = R^*$.

The proof is provided in Appendix D.2. The theorem is an oracle-rate population statement: unbiasedness holds when the channel rates used in the correction equal the true effective rates and before finite-sample GRPO normalization, clipping, or KL terms are applied. Theorem 3.3 shows that, by replacing the noisy reward $\tilde{R}$ with $\widehat{R}$ in any standard policy gradient formulation, we obtain an unbiased estimate of the true policy gradient. This allows $\widehat{R}$ to serve as a drop-in replacement for the reward signal in complex RL algorithms, seamlessly integrating with advantage estimation and other machinery. The details of the algorithm are summarized in Algorithm 1. After rolling out a set of trajectories and obtaining their rewards, we use the corrected rewards to compute the policy gradients for model update.

3.3 Noisy Policy Gradient with Forward Correction (PGFC)

While PGBC provides an elegant unbiased estimator, it faces two practical challenges. First, the denominator $(1 - \rho_0 - \rho_1)$ can be small if the total noise rate is high, leading to high variance in the reward estimate $\widehat{R}$ and potentially unstable training. Second, it requires accurate estimation of both the false positive rate ρ_0 and the false negative rate ρ_1, which are usually difficult to obtain in practice. The PGFC approach is introduced to mitigate these issues.

Instead of correcting the reward itself, this method directly modifies the policy gradient estimator to ensure its expectation is correctly aligned. The core idea is to reweight the score function G_t under

Algorithm 1 Noisy Policy Gradient with Backward Correction (PGBC)

Input: Initial policy θ_0; learning rate η; batch size M; estimates of noise rates, $(\hat{\rho}_0, \hat{\rho}_1)$.
loop
Roll out M trajectories $\{(x_i, y_i)\}_{i=1}^M$, obtaining observed rewards $\{\tilde{R}_i\}_{i=1}^M$.
For each trajectory i, compute the unbiased reward estimate:

$$\widehat{R}_i \leftarrow \frac{\tilde{R}_i - \hat{\rho}_0}{1 - \hat{\rho}_0 - \hat{\rho}_1}.$$

Compute the policy gradient using the corrected rewards $\{\widehat{R}_i\}_{i=1}^M$.
Update parameters: $\theta \leftarrow \theta + \eta\, \Delta\theta$.
end loop

Algorithm 2 Noisy Policy Gradient with Forward Correction (PGFC)

Input: Initial policy θ_0; learning rate η; batch size M; an estimate of the false negative rate, $\hat{\rho}_1$.
loop
Roll out M trajectories and observe rewards $\{\tilde{R}_i\}_{i=1}^M$.
For each trajectory i, define the weight based on the observed reward $\tilde{R}_i$:

$$w_{\tilde{R}_i} \leftarrow \begin{cases} \hat{\rho}_1 - 1, & \text{if } \tilde{R}_i = 0, \\ \hat{\rho}_1, & \text{if } \tilde{R}_i = 1. \end{cases}$$

For each trajectory i, compute the score-function term G_i and the weighted term $h_i \leftarrow w_{\tilde{R}_i} G_i$.
Form the policy gradient estimate: $\hat{g} \leftarrow \frac{1}{M} \sum_{i=1}^M h_i$.
Update parameters: $\theta \leftarrow \theta + \eta\, \hat{g}$.
end loop

the t-th token based on the observed noisy reward $\tilde{R}$. We define an update term $h_t = w_{\tilde{R}} G_t$ and choose the *forward weights* w_0 (for $\tilde{R} = 0$) and w_1 (for $\tilde{R} = 1$) such that the expected update $\mathbb{E}[h_t]$ is parallel to the clean gradient $\nabla_\theta J(\theta)$. The key insight is to choose the forward weights so that the conditional expectations given the true reward R^* have a specific structure.

Proposition 3.4 (Conditional Expectation of Forward Weights)**.** *Let the forward weights be defined as $w_0 = \rho_1 - 1$ and $w_1 = \rho_1$. Under the Verifier Reward Channel model, the conditional expectations of the corresponding weight $w_{\tilde{R}}$ given the true reward R^* are:*

1. $\mathbb{E}[w_{\tilde{R}} \mid R^* = 1] \;=\; 0.$
2. $\mathbb{E}[w_{\tilde{R}} \mid R^* = 0] \;=\; -(1 - \rho_0 - \rho_1).$

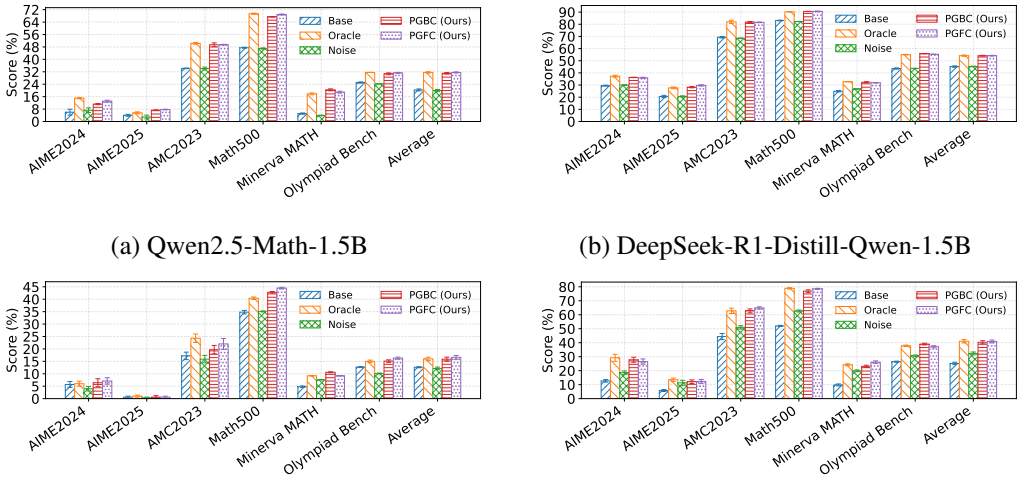
The proof can be found in Appendix D.3. Proposition 3.4 reveals that, in expectation, the forward weights are chosen so that their conditional expectation vanishes when $R^* = 1$, i.e., truly positive cases do not require correction. Consequently, the explicit contribution to the correction term arises from samples with $R^* = 0$. This structure allows us to recover the correct gradient direction, as shown in the following theorem.

Theorem 3.5 (REINFORCE-Style Directional Alignment of the Forward Correction)**.** *Let the gradient-update term be $\Delta\theta = \frac{1}{M} \sum w_{\tilde{R}}\, G_t$, where $w_0 = \rho_1 - 1$ and $w_1 = \rho_1$ with M trajectories. Under the Verifier Reward Channel model, the expected update is parallel to the clean policy gradient:*

$$\mathbb{E}[\Delta\theta] \;=\; c\, \nabla_\theta J(\theta), \tag{4}$$

in which $c = (1 - \rho_0 - \rho_1)$.

The proof is given in Appendix D.4. Since $1 - \rho_0 - \rho_1 > 0$, Theorem 3.5 guarantees that the expected update of the policy is in the same direction as the true gradient; meanwhile, the positive scaling factor $(1 - \rho_0 - \rho_1)$ can be absorbed into the learning rate. By avoiding the inverse operation of PGBC, PGFC circumvents the variance-inflation problem.



(a) Qwen2.5-Math-1.5B

(b) DeepSeek-R1-Distill-Qwen-1.5B

(c) Llama-3.2-3B-Instruct

(d) Qwen2.5-Math-7B

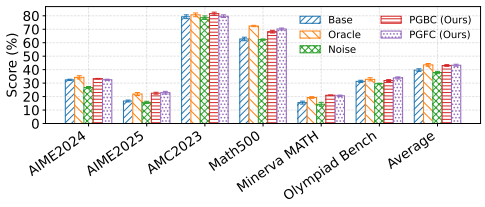
Figure 2: Synthetic-Noise Results **(pass@1)** with 16 samples and 5 random seeds. **Base**: baseline without RL; **Oracle**: Training with clean rewards; **Noise**: Training with noisy verifier rewards; **Noise_BC**: Training with noise under backward correction; **Noise_FC**: Training with noise under forward correction.

In practice, we estimate $\hat{\rho}_1$ during training by *appealing* a small, uniformly random subset of rule-based negatives to a lightweight LLM verifier (e.g., TinyV (Xu et al., 2025)) and then smoothing the empirical flip rate. Concretely, at step t, let $\mathcal{N}_{\mathrm{R}}^{(t)}$ be items labeled negative by the rule-based verifier, $\mathcal{P}_{\mathrm{R}}^{(t)}$ the rule-based positives, and sample a fraction $q \in (0, 1]$ of $\mathcal{N}_{\mathrm{R}}^{(t)}$ for appeal; denote by $\mathcal{P}_{\mathrm{L}}^{(t)}$ those appealed items that the LLM flips to positive. Using a Horvitz–Thompson correction (Karwa & Airoldi, 2023), we estimate the number of FNs as $|\mathcal{P}_{\mathrm{L}}^{(t)}|/q$, treat $|\mathcal{P}_{\mathrm{R}}^{(t)}|$ as true positives in the rule-checker regime, and set $\hat{\rho}_1^{(t)} = \frac{|\mathcal{P}_{\mathrm{L}}^{(t)}|/q+\alpha}{|\mathcal{P}_{\mathrm{L}}^{(t)}|/q+|\mathcal{P}_{\mathrm{R}}^{(t)}|+\alpha+\beta}$ with small Beta-prior pseudocounts $\alpha, \beta = 10^{-5}$. We then apply exponential moving average (EMA) smoothing over a sliding window. If both $\hat{\rho}_0$ and $\hat{\rho}_1$ are available, one can use PGBC; if only $\hat{\rho}_1$ is available, PGFC is the default. We released our code in an anonymous link `https://anonymous.4open.science/r/noisy-RLVR-4BBF/README.md`. More details of appealing mechanism can be found in Appendix E.4.

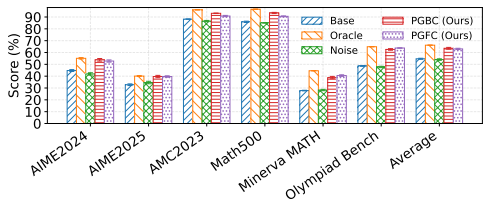
Bridging REINFORCE-Style Corrections to GRPO Updates Our analysis is stated for the reward-gradient component of a score-function estimator, while GRPO implements this component through scalar trajectory coefficients broadcast as token-level advantages inside a clipped surrogate. We therefore insert each correction at the coefficient level that best matches its estimator: PGBC is implemented as a corrected-utility coefficient based on $\widehat{r}_i = (\tilde{r}_i - \hat{\rho}_0)/(1 - \hat{\rho}_0 - \hat{\rho}_1)$, whereas PGFC is implemented as a score-level coefficient $w_{\tilde{r}_i} \in \{\hat{\rho}_1 - 1, \hat{\rho}_1\}$, not as a reward proxy. In both cases, the corrected coefficient is passed directly to the GRPO surrogate, with the KL term unchanged. This implementation avoids a subtle binary-reward degeneracy: any pre-normalization proxy affine in $\tilde{r}_i$ can be removed by group centering and standardization, making the corrected update identical to uncorrected noisy-reward GRPO. Appendix A states the centered-update guarantee and Appendix B gives the implementation recipe.

4 Experiments

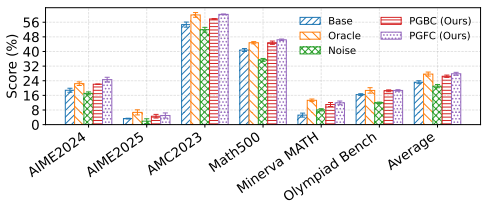
We evaluate our approach under both *synthetic* and *real-world* verifier noise. We first spell out the experimental protocol, and then present main and ablation results. Additional experimental results, including training dynamics, pass@8 performance under real-world noise, appeals statistics, complementarity with stronger verifiers, and non-math evaluations are provided in Appendix C.



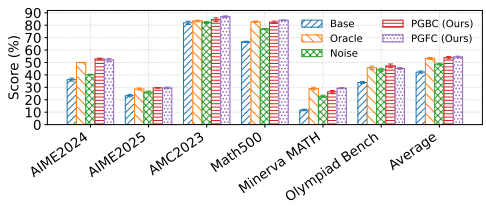
(a) Qwen2.5-Math-1.5B



(b) DeepSeek-R1-Distill-Qwen-1.5B



(c) Llama-3.2-3B-Instruct



(d) Qwen2.5-Math-7B

Figure 3: Synthetic-Noise Results **(pass@8)** with 16 samples and 5 random seeds. **Base**: baseline without RL; **Oracle**: Training with clean rewards; **Noise**: Training with noisy verifier rewards; **Noise_BC**: Training with noise under backward correction; **Noise_FC**: Training with noise under forward correction.

4.1 Experimental Setup

We train on three small backbones, *Qwen2.5-Math-1.5B*, *DeepSeek-R1-Distill-Qwen-1.5B*, and *Llama-3.2-3B-Instruct*, and probe scale with *Qwen2.5-Math-7B*. Rewards come from either (i) a rule-based checker that extracts the final `\boxed{\cdot}` answer and tests numeric/rational equivalence or (ii) a lightweight LLM verifier (**TinyV 1.5B**) used for appeals/estimation; prior work motivates explicit noise modeling due to systematic FP/FN behavior in these verifiers. Unless stated otherwise, we follow the `DeepScaleR` corpus and a GRPO-style on-policy recipe with BoN sampling, implementing our *backward* and *forward* corrections as drop-in hooks at the advantage-construction stage within `VERL`. Evaluation uses six verifiable math suites—AIME-2024, AIME-2025, AMC-2023, MATH500, Minerva MATH, and OlympiadBench—reporting *Pass@1* and *Pass@8* with 16 samples and 5 random seeds in average. Compute is 8×A100 (40GB) GPUs servers; unless noted, KL schedules, sampling temperatures, and other rollout settings are kept identical across compared conditions. We estimate pass@k as $1 - \binom{n-c_{\text{ok}}}{k} / \binom{n}{k}$, where $n = 16$, c_{ok} is the number of correct samples, and $k \in \{1, 8\}$ (Chen et al., 2021). More details can be found in Appendix E.

4.2 Synthetic Noise

To disentangle optimization effects from verifier unreliability, we inject *verifier-conditional* noise into the binary reward stream during training. Concretely, when the latent clean reward is $R^* \in \{0, 1\}$, the observed reward $\tilde{R}$ is drawn from a reward channel with $\Pr(\tilde{R}{=}1 \mid R^*{=}0) = \rho_0$ and $\Pr(\tilde{R}{=}0 \mid R^*{=}1) = \rho_1$; unless stated otherwise we use $\rho_0{=}0.1$ and $\rho_1{=}0.2$. We train GRPO on DeepScaleR with identical sampling and KL settings across conditions, and evaluate Pass@1 (16 samples) and Pass@8 on AIME-2024/2025, AMC-2023, MATH500, Minerva MATH, and OlympiadBench. We compare five variants: *Base* (no RL), *Oracle* (clean rewards), *Noise* (uncorrected), and our two corrections—*PGBC* (backward correction using $\widehat{R} = \frac{\tilde{R}-\rho_0}{1-\rho_0-\rho_1}$) and *PGFC* (forward correction that rescales the gradient using ρ_1 only). Figure 3 complements the pass@1 results with synthetic-noise pass@8: on the Qwen and Llama backbones, PGBC and PGFC improve over Noise and remain close to Oracle in the six-benchmark average.

Table 1: Real-world pass@1. `Rule`: uncorrected rule-based rewards; `LLM-as-Judge`: direct LLM-judge rewards; `Appeals`: rule-based reward plus LLM appeals on negative samples without gradient correction; `Appeals+PGFC`: appeals-based FN-rate estimation plus forward correction.

Dataset	AIME2024	AIME2025	AMC2023	MATH500	Minerva MATH	Olympiad Bench	Average
Qwen2.5-Math-1.5B							
Base	6.0 ± 1.9	4.0 ± 0.6	34.2 ± 0.2	47.5 ± 0.3	5.1 ± 0.4	25.1 ± 0.4	20.3 ± 0.6
Rule	15.0 ± 0.4	5.6 ± 0.6	50.3 ± 0.6	$\mathbf{69.4 \pm 0.4}$	$\mathbf{17.8 \pm 0.6}$	31.6 ± 0.0	31.6 ± 0.4
LLM-as-Judge	10.9 ± 1.3	4.7 ± 1.0	42.1 ± 1.8	63.0 ± 0.7	15.9 ± 0.7	25.3 ± 0.5	27.0 ± 1.0
Appeals	11.9 ± 0.6	5.8 ± 1.2	47.8 ± 1.2	68.3 ± 0.1	16.7 ± 0.6	29.8 ± 0.1	30.1 ± 0.6
Appeals+PGFC (Ours)	$\mathbf{20.3 \pm 0.0}$	$\mathbf{10.7 \pm 1.7}$	$\mathbf{53.3 \pm 1.4}$	68.6 ± 0.8	16.5 ± 0.4	$\mathbf{32.9 \pm 0.2}$	$\mathbf{33.7 \pm 0.8}$
DeepSeek-R1-Distill-Qwen-1.5B							
Base	29.5 ± 0.4	20.6 ± 0.9	69.3 ± 0.6	83.1 ± 0.3	24.9 ± 0.6	43.6 ± 0.5	45.2 ± 0.6
Rule	31.0 ± 0.7	26.3 ± 0.5	79.1 ± 0.9	88.7 ± 0.4	29.5 ± 0.1	50.1 ± 0.2	50.8 ± 0.5
LLM-as-Judge	30.2 ± 1.0	27.0 ± 0.8	79.5 ± 0.9	85.8 ± 0.7	27.7 ± 0.9	47.2 ± 1.0	49.6 ± 0.9
Appeals	34.0 ± 1.2	27.8 ± 0.9	78.7 ± 1.1	88.0 ± 0.8	29.0 ± 1.0	49.6 ± 1.1	51.2 ± 1.0
Appeals+PGFC (Ours)	$\mathbf{35.8 \pm 0.7}$	$\mathbf{29.1 \pm 0.8}$	$\mathbf{81.7 \pm 0.6}$	$\mathbf{88.8 \pm 0.5}$	$\mathbf{31.1 \pm 0.7}$	$\mathbf{52.4 \pm 0.6}$	$\mathbf{53.2 \pm 0.7}$
Llama-3.2-3B-Instruct							
Base	5.7 ± 1.2	0.6 ± 0.4	17.2 ± 1.5	34.8 ± 0.7	4.8 ± 0.3	12.7 ± 0.2	12.6 ± 0.2
Rule	6.0 ± 1.0	$\mathbf{1.0 \pm 0.4}$	$\mathbf{24.3 \pm 1.7}$	40.4 ± 0.5	8.9 ± 0.7	15.0 ± 0.5	15.9 ± 0.8
LLM-as-Judge	1.4 ± 1.1	0.1 ± 0.3	12.7 ± 2.1	26.4 ± 0.2	4.1 ± 0.2	7.7 ± 0.4	8.7 ± 0.6
Appeals	3.8 ± 1.4	0.6 ± 0.7	16.9 ± 1.8	40.8 ± 0.3	9.0 ± 0.4	13.7 ± 0.3	14.1 ± 0.4
Appeals+PGFC (Ours)	$\mathbf{9.4 \pm 1.9}$	0.4 ± 0.6	21.2 ± 1.3	$\mathbf{46.4 \pm 0.3}$	$\mathbf{9.9 \pm 0.8}$	$\mathbf{16.3 \pm 0.3}$	$\mathbf{17.3 \pm 0.9}$
Qwen2.5-Math-7B							
Base	12.7 ± 0.9	5.8 ± 0.7	44.4 ± 2.2	52.0 ± 0.4	9.8 ± 0.7	26.4 ± 0.4	25.2 ± 0.9
Rule	29.2 ± 2.5	13.5 ± 1.2	62.8 ± 1.9	78.9 ± 0.5	24.2 ± 0.7	37.8 ± 0.5	41.1 ± 1.2
LLM-as-Judge	16.8 ± 2.1	6.8 ± 1.4	50.2 ± 1.8	62.5 ± 1.0	10.1 ± 0.4	31.0 ± 0.9	29.6 ± 1.3
Appeals	27.4 ± 1.9	11.8 ± 1.9	63.7 ± 1.6	74.9 ± 0.7	20.6 ± 0.8	37.6 ± 0.3	39.3 ± 1.2
Appeals+PGFC (Ours)	$\mathbf{31.0 \pm 1.5}$	$\mathbf{14.6 \pm 1.2}$	$\mathbf{65.7 \pm 1.4}$	$\mathbf{81.6 \pm 0.4}$	$\mathbf{26.2 \pm 0.5}$	$\mathbf{39.3 \pm 0.5}$	$\mathbf{43.1 \pm 0.9}$

4.3 Real-World Noise

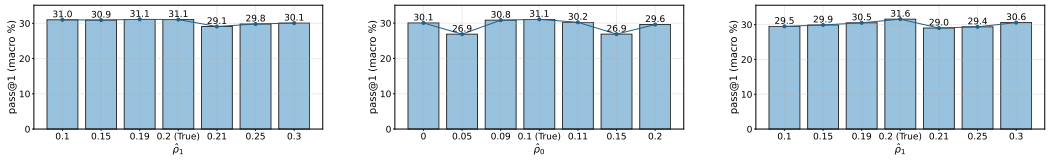
As discussed above, automated verifiers exhibit both *false positives* (LLM judges over-crediting incorrect solutions) and *false negatives* (rule-based checkers rejecting correct ones) (Gu et al., 2024; Xu et al., 2025; Huang et al., 2025; Li et al., 2025). In math RL pipelines, the latter is particularly prevalent: brittle exact-match or limited equivalence rules lead to many valid answers being scored as incorrect, depriving the agent of learning signal. Motivated by this, in this subsection we specifically investigate whether denoising *false negatives*, i.e., estimating $\hat{\rho}_1$ as in Section 3.3, can measurably improve policy optimization. Concretely, we keep a fast rule-based checker as the *primary* reward source with a lightweight LLM verifier (TinyV) (Xu et al., 2025). From disagreements on negatives we maintain an online estimate $\hat{\rho}_1$ (EMA over a sliding window to track policy drift).

Results in Table 1 show a consistent pattern across backbones and benchmarks. Direct LLM-judge rewards underperform Rule on the six-benchmark average for each backbone, although individual task outcomes vary. Employing appeals to recheck rule-based negatives (`Appeals`) recovers some FNs, but its downstream benefit varies across models and benchmarks. In contrast, `Appeals+PGFC` uses the same appeals stream to estimate $\hat{\rho}_1$ and directly applies the FN correction to the policy gradient via weights $w_{\tilde{R}} \in \{\hat{\rho}_1 - 1, \hat{\rho}_1\}$, improving over Appeals in 22 of 24 cells.

4.4 Robustness to Noise Mis-Specification

There now exist practical procedures to *estimate* verifier FP/FN rates via rule-based equivalence checkers (e.g., math-verify), lightweight appeals/judges such as TinyV, and meta-evaluation suites that quantify verifier reliability (Hugging Face, 2025; Xu et al., 2025; Li et al., 2025). In real deployments, however, these estimates can be imperfect. We therefore test how *backward* and *forward* corrections behave under mis-specified noise rates. We follow the synthetic-noise setting from Section 4.2 with Qwen2.5-math-1.5B: rewards are *corrupted at data-time* with $\rho_0{=}0.1,\ \rho_1{=}0.2$. During training, we *intentionally* feed each algorithm mis-specified rates from a grid $\hat{\rho}_0 \in [0.0, 0.2]$, $\hat{\rho}_1 \in [0.1, 0.3]$, and report the Average Pass@1 (16 samples) across our six math benchmarks. Because the forward method only requires the false negative rate, we vary $\hat{\rho}_1$ for forward correction while keeping $\hat{\rho}_0$ unused, as shown in Fig. 4.

Backward correction remains strong when the total noise is underestimated, but performance degrades as we overestimate the rates (i.e., as $1 - \hat{\rho}_0 - \hat{\rho}_1$ shrinks). Intuitively, the unbiased de-biasing,



(a) BC: $\hat{\rho}_0$=0.1, $\hat{\rho}_1 \in [0.1, 0.3]$ (b) BC: $\hat{\rho}_1$=0.2, $\hat{\rho}_0 \in [0.0, 0.2]$ (c) FC: $\hat{\rho}_0$=0.1, $\hat{\rho}_1 \in [0.1, 0.3]$

Figure 4: Robustness results. (a) Backward correction (BC) with $\hat{\rho}_0$ fixed and sweeping $\hat{\rho}_1$; (b) Backward correction (BC) with $\hat{\rho}_1$ fixed and sweeping $\hat{\rho}_0$; (c) Forward correction (FC) with $\hat{\rho}_0$ fixed and sweeping $\hat{\rho}_1$.

Table 2: Format-dependent **non-iid verifier-noise** setting on Qwen2.5-Math-1.5B. Each cell reports pass@1 / pass@8.

Method	MATH500	AIME2024	AIME2025	AMC2023	Minerva Math	OlympiadBench	Average
Base	47.5 / 62.8	6.0 / 32.4	4.0 / 16.7	34.2 / 79.3	5.1 / 15.5	25.1 / 31.3	20.3 / 39.7
Noise	40.7 / 60.5	7.5 / 26.5	2.2 / 15.8	30.0 / 72.4	3.8 / 11.6	21.5 / 25.8	17.6 / 35.4
PGBC	67.7 / 67.9	11.5 / **33.0**	7.4 / 22.4	49.1 / **81.8**	20.7 / 21.1	30.5 / 31.7	31.2 / 43.0
PGFC	**68.1 / 69.9**	**12.8** / 32.4	**8.0 / 22.6**	**49.6** / 79.5	**21.8 / 23.8**	**31.1 / 33.6**	**31.9 / 43.6**

$\widehat{R} = \frac{\tilde{R} - \hat{\rho}_0}{1 - \hat{\rho}_0 - \hat{\rho}_1}$, involves an inverse factor that amplifies reward variance and accentuates estimation error when the effective denominator becomes small. In contrast, *forward* correction exhibits a notably flat response curve around the true ρ_1; perturbing $\hat{\rho}_1$ changes only the per-sample weights and avoids any inverse operation, yielding lower-variance policy gradient estimates.

We also stress-test the constant-channel assumption using format-dependent, **non-iid verifier noise** with the same Qwen2.5-Math-1.5B / DeepScaleR setup. Both the FN and FP rates depend on a prompt-level tag and a response-level format tag (e.g., decimal or missing boxed answer vs. canonical format). Specifically, we use a shared FP rate of $\rho_0 = 0.2$ but a format-dependent FN rate: $\rho_1 = 0.05$ for low-risk formats and $\rho_1 = 0.45$ for high-risk formats, with the actual flip decision made by hashing `(data_source, ground_truth, solution_str)`. This explicitly violates the instance-independent channel model. Table 2 shows that both corrections still improve over uncorrected training and base model under this non-iid noise setting, with PGFC again strongest performance in most cases.

5 Conclusion

Reinforcement Learning with Verifiable Rewards (RLVR) has emerged as a practical and popular paradigm for training reasoning-focused large language models (LLMs). Yet binarized rewards inevitably suffer from *false positives* (FP) and *false negatives* (FN). Prior efforts largely mitigated these errors from the verifier side—e.g., debiasing LLM judges or strengthening equivalence checking—to reduce the downstream impact on learning (Li et al., 2025; Huang et al., 2025). In contrast, we modeled verifier unreliability as a verifier-conditional noise channel with the FP and FN rates, and introduced two theory-driven corrections that act directly at the policy-learning interface. The *backward* correction de-biases the observed reward to yield an unbiased gradient estimator; it requires estimates of the FP and FN rates but is agnostic to the choice of policy-optimization algorithm and can thus be used beyond policy gradient methods. The *forward* correction rescales score-function terms so that the expected update is aligned with the clean gradient; notably, it needs only the FN rate estimate, which is often the dominant source of real-world noise due to rule-based FNs (Huang et al., 2025; Xu et al., 2025).

While we instantiated appeals with a lightweight LLM verifier for efficiency, the same backward/forward correction mechanisms should also gain from stronger verifiers (Xu et al., 2025). Meanwhile, although modeled with the constant noise rates, the proposed methods already show clear advantages, even under a more challenging non-iid verifier noise setting. On the other hand, real-world noise can depend on content, formatting, prompt style, and verifier family, motivating future work on instance-dependent reliability models and adversarially robust verifiers.

AI USE DISCLOSURE

We used AI tools to polish the language of the manuscript.

REFERENCES

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or llms as the judge? a study on judgement bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8301–8327, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 3828–3850. Association for Computational Linguistics, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Yuzhen Huang, Weihao Zeng, Xingshan Zeng, Qi Zhu, and Junxian He. Pitfalls of rule- and model-based verifiers–a case study on mathematical reasoning. *arXiv preprint arXiv:2505.22203*, 2025.

Hugging Face. Math-verify: A robust mathematical expression evaluation system designed for assessing large language model outputs in mathematical tasks. GitHub repository, 2025. URL https://github.com/huggingface/Math-Verify.

HuggingFaceH4. Aime 2024 (dataset card). Hugging Face, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pp. 2304–2313. PMLR, 2018.

Vishesh Karwa and Edoardo M Airoldi. On the admissibility of horvitz-thompson estimator for estimating causal effects under network interference. *arXiv preprint arXiv:2312.01234*, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. 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 Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Junnan Li, Richard Socher, and Steven C. H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Xuefeng Li, Tongliang Liu, Bo Han, Gang Niu, and Masashi Sugiyama. Provably end-to-end label-noise learning without anchor points. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6403–6413. PMLR, 2021.

Xuzhao Li, Xuchen Li, Shiyu Hu, Yongzhen Guo, and Wentao Zhang. Verifybench: A systematic benchmark for evaluating reasoning verifiers across domains. *arXiv preprint arXiv:2507.09884*, 2025.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Shudong Liu, Hongwei Liu, Junnan Liu, Linchen Xiao, Songyang Gao, Chengqi Lyu, Yuzhe Gu, Wenwei Zhang, Derek F. Wong, Songyang Zhang, and Kai Chen. CompassVerifier: A unified and robust verifier for LLMs evaluation and outcome reward. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33466–33494, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1698. URL https://aclanthology.org/2025.emnlp-main.1698/.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

math-ai. Amc 2023 (dataset card). Hugging Face, 2025. URL https://huggingface.co/datasets/math-ai/amc23.

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

Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 1196–1204, 2013.

OpenCompass. Aime 2025 (dataset card). Hugging Face, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2233–2241. IEEE Computer Society, 2017.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. 2024. URL https://arxiv.org/abs/2402.03300.

Jiawen Shi, Zenghui Yuan, Yinuo Liu, Yue Huang, Pan Zhou, Lichao Sun, and Neil Zhenqiang Gong. Optimization-based prompt injection attack to llm-as-a-judge. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie (eds.), *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pp. 660–674. ACM, 2024.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 292–314, 2025.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE transactions on neural networks and learning systems*, 34(11):8135–8153, 2022.

Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller (eds.), *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1057–1063. The MIT Press, 1999.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges. *arXiv preprint arXiv:2406.12624*, 2024.

Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 6202–6209. AAAI Press, 2020.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9*, 2022.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

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

Zhangchen Xu, Yuetai Li, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. Tinyv: Reducing false negatives in verification improves rl for llm reasoning. *arXiv preprint arXiv:2505.14625*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Yulai Zhao, Haolin Liu, Dian Yu, Sunyuan Kung, Meijia Chen, Haitao Mi, and Dong Yu. One token to fool llm-as-a-judge. *arXiv preprint arXiv:2507.08794*, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

A FROM REINFORCE-STYLE CORRECTIONS TO GRPO UPDATES

Setup (single prompt and group). Fix a prompt x and sample a group of K i.i.d. responses $y_1, \dots, y_K \sim \pi_\theta(\cdot \mid x)$. Let $G_i \triangleq \sum_{t=1}^{|y_i|} \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})$ denote the trajectory score-function. Let u_i be any scalar *proxy reward* computed from the trajectory. For PGBC, $u_i = \widehat{r}_i$. For the population PGFC theorem, the score-function term is weighted as $w_{\tilde{r}_i} G_i$; the hypothetical pre-centering proxy $u_i = \tilde{r}_i w_{\tilde{r}_i}$ is analyzed below only as a degeneracy example, not as our default implementation. Define the group mean $\bar{u} = \frac{1}{K}\sum_{j=1}^{K} u_j$. Consider the centered score-function update

$$\Delta_{\text{center}}(\theta) \;\triangleq\; \frac{1}{K}\sum_{i=1}^{K}(u_i - \bar{u})\, G_i. \tag{5}$$

Proposition A.1 (Group centering preserves expected direction)**.** *Assume $\{(u_i, G_i)\}_{i=1}^{K}$ are i.i.d. under $\pi_\theta(\cdot \mid x)$ and $\mathbb{E}[G_i \mid x] = 0$ (the standard score-function identity (Williams, 1992; Sutton et al., 1999)). Then the centered update in* (5) *satisfies*

$$\mathbb{E}[\Delta_{\textit{center}}(\theta) \mid x] \;=\; \Big(1 - \frac{1}{K}\Big)\, \mathbb{E}[u\, G \mid x]\,. \tag{6}$$

In particular, if $\mathbb{E}[u\, G \mid x]$ is parallel to the clean gradient direction for the prompt-conditioned objective, then so is $\mathbb{E}[\Delta_{\textit{center}}(\theta) \mid x]$.

Proof. Expand $\Delta_{\text{center}} = \frac{1}{K}\sum_i u_i G_i - \bar{u} \cdot \frac{1}{K}\sum_i G_i$. Taking expectation conditioned on x yields

$$\mathbb{E}[\bar{u} \cdot \tfrac{1}{K}\sum_i G_i \mid x] = \frac{1}{K^2}\sum_{i,j}\mathbb{E}[u_j G_i \mid x].$$

The $i = j$ terms contribute $\frac{K}{K^2}\mathbb{E}[uG \mid x]$. For $i \neq j$, independence gives $\mathbb{E}[u_j G_i \mid x] = \mathbb{E}[u \mid x]\mathbb{E}[G \mid x] = 0$. Thus $\mathbb{E}[\bar{u} \cdot \frac{1}{K}\sum_i G_i \mid x] = \frac{1}{K}\mathbb{E}[uG \mid x]$, implying (6). □

Corollary A.2 (Directional correctness of PGFC under centered GRPO-style updates)**.** *Under the Verifier Reward Channel (Definition 3.1) and the conditional-independence assumption stated above, let $u = w_{\tilde{R}}$ with $w_0 = \rho_1 - 1$ and $w_1 = \rho_1$ as in Proposition 3.4. Then for each prompt x,*

$$\mathbb{E}[\Delta_{\textit{center}}(\theta) \mid x] \;=\; \Big(1 - \frac{1}{K}\Big)(1 - \rho_0 - \rho_1)\, \nabla_\theta J(\theta \mid x),$$

so the expected centered update remains aligned with the clean policy-gradient direction.

Binary-proxy equivalence caveat. The corollary explains how centering interacts with a generic proxy whose score-function expectation is aligned with the clean gradient. It does not by itself prove that every PGFC proxy creates a distinct GRPO update. For binary rewards, $w_{\tilde{r}} = \tilde{r} + \rho_1 - 1$ differs from $\tilde{r}$ by an additive constant, so group centering removes the difference. Likewise, the hypothetical pre-centering proxy $u = \tilde{r}\, w_{\tilde{r}} = \rho_1 \tilde{r}$ is a positive rescaling of $\tilde{r}$; if within-group standardization is enabled, this rescaling also cancels.

Standardization, clipping, and KL (what is guaranteed, and what is approximate). GRPO further divides centered proxy rewards by the within-group standard deviation to form normalized advantages, and then applies a PPO-style clipping on token-level ratios and an explicit KL penalty to π_{ref} (Shao et al., 2024). The exact direction-preservation statement in Proposition A.1 covers the *centering* step, which is the main nontrivial departure from vanilla REINFORCE. Clipping and the KL penalty add further stabilizers: in the small-update regime (encouraged by the KL term and typical learning rates), clipping is rarely active and the update is close to the unclipped score-function form; when clipping activates, it intentionally limits step size and may deviate from the exact REINFORCE direction. Importantly, our corrections target the *reward-gradient component*; the KL gradient is unchanged, so any global scaling factor (e.g., $1 - \rho_0 - \rho_1$) can be absorbed into the effective learning rate and/or interpreted as modifying the reward–KL trade-off.

B Implementation Details

This appendix describes the implementation of PGBC and PGFC inside a GRPO-style RLVR pipeline. The key design choice is to insert each correction at the level of the scalar coefficient multiplying the score-function term. PGBC is a *corrected-utility* method: it replaces the observed verifier reward by an estimate of the clean reward. PGFC is a *score-corrected* method: it replaces the usual reward-derived advantage by a signed channel-calibrated multiplier. In both cases, the corrected coefficient is passed directly to the PPO-style GRPO surrogate and is not converted into a binary reward proxy that is subsequently group-standardized.

B.1 Notation and GRPO Surrogate

For a prompt x, GRPO samples a group of K responses

$$y_1, \dots, y_K \sim \pi_{\theta_{\text{old}}}(\cdot \mid x).$$

Let $\tilde{r}_i \in \{0, 1\}$ be the observed verifier reward for response y_i. For token t in response i, define the policy ratio

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

Given a scalar trajectory coefficient a_i, we broadcast it to all response tokens:

$$A_{i,t} \equiv a_i, \qquad t = 1, \dots, |y_i|.$$

The clipped GRPO reward surrogate is then

$$\mathcal{L}^{\text{rew}}_{\text{GRPO}}(\theta) = \frac{1}{K} \sum_{i=1}^{K} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min(r_{i,t}(\theta) A_{i,t}, \text{clip}(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon) A_{i,t}) \,.$$

The full objective keeps the usual KL penalty:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathcal{L}^{\text{rew}}_{\text{GRPO}}(\theta) - \beta \, \text{KL}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)) \,.$$

At $\theta = \theta_{\text{old}}$, when clipping is inactive, the reward-gradient component reduces to the score-function update

$$\nabla_\theta \mathcal{L}^{\text{rew}}_{\text{GRPO}}(\theta) = \frac{1}{K} \sum_{i=1}^{K} a_i \left(\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t}) \right),$$

up to the conventional length normalization used by the implementation. Therefore, the scalar coefficient a_i is the correct place to insert PGBC or PGFC.

B.2 PGBC: Corrected-Utility GRPO

PGBC estimates the clean binary reward by inverting the verifier channel. Given estimates $\hat{\rho}_0, \hat{\rho}_1$, define

$$\hat{c} = 1 - \hat{\rho}_0 - \hat{\rho}_1.$$

We require $\hat{c} > 0$ and in code clamp it below by a small constant $\epsilon_c > 0$ for numerical safety. The corrected utility is

$$\widehat{r}_i = \frac{\tilde{r}_i - \hat{\rho}_0}{\hat{c}}.$$

This quantity is used as the trajectory coefficient, optionally with a variance-reduction baseline:

$$a_i^{\text{BC}} = \frac{\widehat{r}_i - b_i^{\text{BC}}}{s_{\text{BC}}}.$$

Here $s_{\text{BC}} > 0$ is a fixed or slowly moving positive scale, such as an EMA scale, and should not be the current group's standard deviation. The default baseline is leave-one-out:

$$b_i^{\text{BC}} = \frac{1}{K-1} \sum_{j \neq i} \widehat{r}_j \qquad (K > 1),$$

with $b_i^{\mathrm{BC}} = 0$ when $K = 1$. The leave-one-out baseline is independent of G_i conditional on x, so before clipping and KL it preserves the expected score-function update:

$$\mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}(\widehat{r}_i - b_i^{\mathrm{BC}})G_i \,\middle|\, x\right] = \mathbb{E}[\widehat{r}\,G \mid x].$$

When the channel rates are correct, $\mathbb{E}[\widehat{r}\,G \mid x]$ equals the clean reward-gradient component. Using the group mean including the i-th sample is also possible, but it introduces the familiar multiplicative factor $1 - 1/K$.

The corrected coefficient a_i^{BC} is then broadcast to all tokens:

$$A_{i,t}^{\mathrm{BC}} \equiv a_i^{\mathrm{BC}}.$$

No current-group z-score standardization is applied to $\{\widehat{r}_i\}_{i=1}^{K}$, because such standardization would remove the affine calibration induced by PGBC in the binary-reward case.

B.3 PGFC: Score-Corrected GRPO

PGFC is implemented as a score-level correction rather than as a reward proxy. Given an estimate $\hat{\rho}_1$, define the signed score multiplier

$$w_{\tilde{r}_i} = \tilde{r}_i - (1 - \hat{\rho}_1) = \begin{cases} \hat{\rho}_1 - 1, & \tilde{r}_i = 0, \\ \hat{\rho}_1, & \tilde{r}_i = 1. \end{cases}$$

The trajectory coefficient is

$$a_i^{\mathrm{FC}} = \frac{w_{\tilde{r}_i} - b_i^{\mathrm{FC}}}{s_{\mathrm{FC}}}.$$

Our default is

$$b_i^{\mathrm{FC}} = 0.$$

The reason is that subtracting the current group mean of $w_{\tilde{r}_i}$ would reduce PGFC to an ordinary relative noisy-reward update, because $w_{\tilde{r}_i}$ is an affine function of $\tilde{r}_i$. A fixed or historical baseline independent of the current sampled actions may be used, but current-group centering is not the default.

The scale $s_{\mathrm{FC}} > 0$ controls the reward–KL trade-off. For pure score-function direction, any fixed positive scale preserves the direction. When both noise rates are available, one may set

$$s_{\mathrm{FC}} = 1 - \hat{\rho}_0 - \hat{\rho}_1$$

to approximately match the clean reward-gradient magnitude. In the FN-dominant rule-checker regime where $\rho_0 \approx 0$, a practical choice is

$$s_{\mathrm{FC}} = 1 - \hat{\rho}_1.$$

If no calibrated scale is desired, we set $s_{\mathrm{FC}} = 1$ and tune β or the learning rate as usual.

The token-level advantage is

$$A_{i,t}^{\mathrm{FC}} \equiv a_i^{\mathrm{FC}}.$$

Before clipping and KL, if $\hat{\rho}_1 = \rho_1$, the population score-function component satisfies

$$\mathbb{E}[a_i^{\mathrm{FC}} G_i \mid x] = \frac{1 - \rho_0 - \rho_1}{s_{\mathrm{FC}}} \nabla_\theta J(\theta \mid x).$$

Thus PGFC preserves the clean reward-gradient direction and, with a suitable positive scale, can also calibrate the reward-gradient magnitude. The KL gradient is not corrected; therefore s_{FC} and β jointly determine the effective reward–KL trade-off.

Remark on the role of $\hat{\rho}_1$. The theoretical conditional-cancellation identity for PGFC uses the true ρ_1. In an unclipped population score-function estimator, adding any action-independent constant to $\tilde{r}_i$ has zero expected gradient by the score-function identity. However, in finite-batch GRPO with clipping, KL regularization, length normalization, and nonstationary rollouts, the fixed channel baseline $1 - \hat{\rho}_1$ affects the realized signed advantages and the clipped surrogate.

B.4 Why We Do Not Use Direct-Group z-Scoring for Binary Channel Corrections

Let $Z_i(u)$ denote the standard group-normalized advantage

$$Z_i(u) = \frac{u_i - \bar{u}}{\mathrm{std}(\{u_j\}_{j=1}^K) + \epsilon}, \qquad \bar{u} = \frac{1}{K}\sum_{j=1}^K u_j.$$

If $u_i = a\tilde{r}_i + b$ with $a > 0$, then

$$u_i - \bar{u} = a(\tilde{r}_i - \bar{\tilde{r}}), \qquad \mathrm{std}(\{u_j\}) = a\ \mathrm{std}(\{\tilde{r}_j\}),$$

and therefore

$$Z_i(u) = \frac{\tilde{r}_i - \bar{\tilde{r}}}{\mathrm{std}(\{\tilde{r}_j\}) + \epsilon/a}.$$

When ϵ is small and the group is non-degenerate, this is numerically identical to the uncorrected noisy-reward advantage.

This degeneracy applies to the most obvious binary implementations of both corrections:

$$\widehat{r}_i = \frac{\tilde{r}_i - \hat{\rho}_0}{1 - \hat{\rho}_0 - \hat{\rho}_1} \quad \text{is affine in } \tilde{r}_i,$$

and

$$w_{\tilde{r}_i} = \tilde{r}_i + \hat{\rho}_1 - 1, \qquad \tilde{r}_i w_{\tilde{r}_i} = \hat{\rho}_1 \tilde{r}_i \quad \text{for } \tilde{r}_i \in \{0, 1\}.$$

Thus, feeding any of these quantities into the usual current-group z-score advantage construction can make the corrected method exactly equivalent to the uncorrected noisy baseline. For this reason, our corrected implementations use the scalar coefficients a_i^{BC} and a_i^{FC} directly, with only fixed or running positive scales.

B.5 Process-Supervised Extension

For process supervision, suppose response y_i has step-level verifier rewards

$$\tilde{r}_i^{(1)}, \ldots, \tilde{r}_i^{(J_i)}$$

attached to step boundaries

$$\tau_i^{(1)} < \cdots < \tau_i^{(J_i)}.$$

PGBC is applied to each step reward:

$$\widehat{r}_i^{(j)} = \frac{\tilde{r}_i^{(j)} - \hat{\rho}_0}{1 - \hat{\rho}_0 - \hat{\rho}_1}.$$

The token coefficient is the return-to-go of corrected utilities:

$$A_{i,t}^{\mathrm{BC}} = \frac{1}{s_{\mathrm{BC}}} \sum_{j:\tau_i^{(j)} \geq t} \left(\widehat{r}_i^{(j)} - b_{i,j}^{\mathrm{BC}} \right),$$

where $b_{i,j}^{\mathrm{BC}}$ is an optional leave-one-out or historical baseline for the corresponding step.

PGFC is applied as a step-level signed score multiplier:

$$w_i^{(j)} = \tilde{r}_i^{(j)} - (1 - \hat{\rho}_1),$$

and the token coefficient is

$$A_{i,t}^{\mathrm{FC}} = \frac{1}{s_{\mathrm{FC}}} \sum_{j:\tau_i^{(j)} \geq t} w_i^{(j)}.$$

As in the outcome case, we avoid current-batch z-scoring of these binary affine quantities. If a positive normalization is needed for numerical stability, we use a fixed or EMA scale rather than the current batch's standard deviation.

B.6 When Conditional Independence Is Violated (Bias Decomposition).

Our directional-alignment results rely on conditional independence to factorize $\mathbb{E}[w_{\tilde{R}} G \mid R^*] = \mathbb{E}[w_{\tilde{R}} \mid R^*]\mathbb{E}[G \mid R^*]$. If verifier errors depend on trajectory features beyond correctness (e.g., style-based FP hacks), the update admits the decomposition

$$\mathbb{E}[w_{\tilde{R}} G \mid R^*] \;=\; \mathbb{E}[w_{\tilde{R}} \mid R^*]\mathbb{E}[G \mid R^*] \;+\; \mathrm{Cov}(w_{\tilde{R}}, G \mid R^*),$$

so the residual bias is governed by the conditional covariance term. This highlights a practical diagnostic: if verifier decisions are strongly predictable from superficial cues that also affect generation likelihood, $\mathrm{Cov}(w_{\tilde{R}}, G \mid R^*)$ may be non-negligible and one should combine channel-based correction with verifier hardening or instance-dependent reliability modeling.

B.7 Online FN-Rate Estimation via Appeals

In the FN-dominant rule-checker regime, PGFC requires an estimate of the false-negative rate ρ_1. At training step t, let

$$\mathcal{N}_{\mathrm{R}}^{(t)} = \{i : \tilde{r}_i = 0\}, \qquad \mathcal{P}_{\mathrm{R}}^{(t)} = \{i : \tilde{r}_i = 1\}$$

be the rule-based negatives and positives. We appeal each rule-negative item independently with probability $q \in (0, 1]$ using a lightweight auxiliary verifier. Let $S^{(t)} \subseteq \mathcal{N}_{\mathrm{R}}^{(t)}$ be the appealed subset, and let

$$\mathcal{P}_{\mathrm{L}}^{(t)} \subseteq S^{(t)}$$

be the appealed items flipped to positive by the auxiliary verifier. The Horvitz–Thompson estimate of the number of false negatives is

$$\widehat{\mathrm{FN}}^{(t)} = \frac{|\mathcal{P}_{\mathrm{L}}^{(t)}|}{q}.$$

Assuming rule-based positives have negligible FP rate, we estimate

$$\hat{\rho}_1^{(t)} = \frac{\widehat{\mathrm{FN}}^{(t)} + \alpha}{\widehat{\mathrm{FN}}^{(t)} + |\mathcal{P}_{\mathrm{R}}^{(t)}| + \alpha + \beta},$$

where $\alpha, \beta > 0$ are small pseudocounts. To track policy drift, we use EMA smoothing:

$$\bar{\rho}_1^{(t)} = \lambda \bar{\rho}_1^{(t-1)} + (1 - \lambda)\hat{\rho}_1^{(t)}.$$

The smoothed value $\bar{\rho}_1^{(t)}$ is used as $\hat{\rho}_1$ in PGFC. The expected number of auxiliary verifier calls per update is

$$q\,|\mathcal{N}_{\mathrm{R}}^{(t)}|.$$

For PGBC, both ρ_0 and ρ_1 are needed. When ρ_0 is unavailable or unreliable, we prefer PGFC in the FN-dominant setting. When both rates can be estimated from a calibration set or from a dual-verifier protocol, PGBC can be used with the corrected utility above.

C Additional Experimental Results

C.1 Performance under Pass@8 Metric

Table 3 retains the measured pass@8 results for the four backbones. The conclusion remains consistent with the pass@1 results: PGBC and PGFC provide reliable improvements over the noisy baseline.

C.2 Complementarity with Stronger Verifiers

We additionally investigate whether the proposed correction mechanisms are synergistic with more capable verifiers. While our main experiments instantiated the appeal process with a lightweight LLM verifier (TinyV) to minimize computational overhead, Table 5 demonstrates that the forward correction yields consistent improvements even when complementing stronger verifier baselines. Specifically, we compare against a "Stronger checker" that extracts the final `\boxed{}`

answer and applies numeric/rational equivalence checking, using the `Math-Verify` library as a robust baseline. Concretely, we parse the gold answer using a gold-appropriate extraction configuration (`ExprExtractionConfig` for plain expressions or `LatexExtractionConfig` for LaTeX), and parse model predictions using `LatexExtractionConfig` together with `ExprExtractionConfig`. Verification is then performed by `Math-Verify`'s parse-and-verify pipeline with reward-verification-oriented normalization. Under this setup, PGFC uses the standard rule-based reward as the primary signal and calls the auxiliary `Math-Verify` checker only on appealed rule-negative samples. With the synthetic noise rates fixed at $\rho_0 = 0.1$ and $\rho_1 = 0.2$ as in the main paper, our `Appeals+PGFC` method achieves notable gains over the strong baseline (e.g., on MATH500 and AMC2023). These results suggest that the theoretical corrections act orthogonally to verifier engineering: regardless of how accurate the underlying verifier is, explicitly modeling and correcting for the residual false-negative rate provides a reliable performance lift.

C.3 Training Dynamics under Synthetic Noise

To better understand how the proposed corrections influence the learning process, we analyze the training dynamics under synthetic i.i.d. noise. As shown in Figure 5, we monitor the group reward standard deviation, normalized advantage standard deviation, and gradient norm across training steps. The purpose of this analysis is to verify that the backward and forward corrections actively modify the advantage construction during training, rather than merely shifting the final endpoint scores. The results demonstrate that both PGBC and PGFC measurably alter these intermediate training statistics relative to the uncorrected noisy baseline, providing evidence that our corrections dynamically adjust the policy gradient throughout the optimization process.

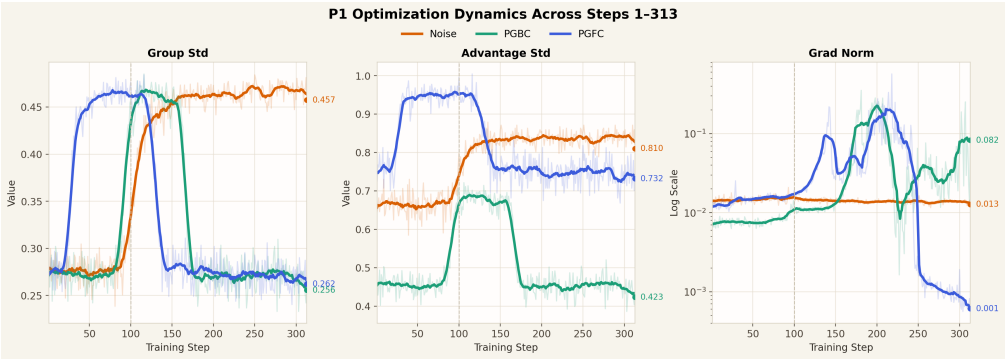


Figure 5: Training dynamics under synthetic iid noise. PGBC and PGFC change the group reward standard deviation, normalized advantage standard deviation, and gradient norm relative to uncorrected noisy training, illustrating that the correction is active inside advantage construction rather than only changing endpoint scores.

C.4 Appeals Statistics and Computational Overhead

To quantify the computational cost and the empirical behavior of the appeal mechanism under real-world verifier noise, we track the appeal and flip statistics during training on *Qwen2.5-Math-1.5B*. As detailed in Table 4, we measure the *appeal rate* (the fraction of rollout responses that trigger an appeal) and the *flip rate* (the fraction of appealed responses that are successfully overturned by the LLM judge). The two methods use the same negative-side appeal mechanism. Within Appeals+PGFC, these calls support both reward recovery and online FN-rate estimation, without a separate post-appeal verifier pass. The total number of auxiliary `TinyV` calls remains small relative to the total number of training rollouts, demonstrating that our online false-negative rate estimation and forward correction are highly computationally efficient.

Table 3: Pass@8 under real-world noise.

Dataset	AIME2024	AIME2025	AMC2023	MATH500	Minerva MATH	Olympiad Bench	Average
Qwen2.5-Math-1.5B							
Base	32.4 ± 0.5	16.7 ± 0.7	79.3 ± 1.4	62.8 ± 1.3	15.5 ± 1.1	31.3 ± 0.8	39.7 ± 1.0
Rule	**34.3 ± 1.2**	19.9 ± 1.1	80.7 ± 1.4	69.6 ± 0.5	18.0 ± 0.6	32.8 ± 1.2	**42.6 ± 1.0**
LLM-as-Judge	29.6 ± 1.1	15.4 ± 0.4	80.0 ± 0.6	63.4 ± 1.2	16.2 ± 0.8	28.6 ± 0.3	38.9 ± 0.7
Appeals	30.5 ± 0.2	**20.5 ± 0.7**	80.7 ± 0.9	68.9 ± 1.3	17.6 ± 1.0	30.5 ± 0.5	41.5 ± 0.8
Appeals+PGFC (Ours)	31.0 ± 0.2	20.0 ± 0.6	**82.2 ± 0.4**	**69.8 ± 1.1**	**18.2 ± 0.5**	**33.3 ± 0.3**	42.4 ± 0.5
DeepSeek-R1-Distill-Qwen-1.5B							
Base	44.8 ± 0.8	32.8 ± 0.7	88.2 ± 0.1	85.9 ± 0.6	27.8 ± 0.3	48.6 ± 0.3	54.7 ± 0.4
Rule	55.1 ± 0.6	40.1 ± 0.4	96.1 ± 0.2	96.6 ± 0.1	44.6 ± 0.1	64.9 ± 0.1	66.2 ± 0.1
LLM-as-Judge	55.4 ± 0.9	39.8 ± 0.7	96.2 ± 0.2	95.7 ± 0.2	43.2 ± 0.7	62.8 ± 0.7	65.5 ± 0.3
Appeals	57.7 ± 1.0	42.5 ± 0.8	96.0 ± 0.2	96.4 ± 0.2	44.2 ± 0.8	64.5 ± 0.8	66.9 ± 0.3
Appeals+PGFC (Ours)	**61.3 ± 0.6**	**44.6 ± 0.7**	**98.6 ± 0.1**	**97.1 ± 0.2**	**46.8 ± 0.6**	**66.1 ± 0.4**	**69.1 ± 0.2**
Llama-3.2-3B-Instruct							
Base	18.8 ± 1.1	3.3 ± 0.2	54.7 ± 1.4	40.8 ± 0.8	5.2 ± 1.1	16.4 ± 0.5	23.2 ± 0.8
Rule	**22.4 ± 1.0**	**6.7 ± 1.4**	**60.0 ± 1.3**	44.8 ± 0.6	**13.3 ± 0.7**	**18.7 ± 1.5**	**27.7 ± 1.1**
LLM-as-Judge	10.0 ± 0.9	0.8 ± 0.9	44.4 ± 1.4	27.2 ± 0.9	6.7 ± 1.4	9.5 ± 0.2	16.4 ± 0.9
Appeals	17.5 ± 0.2	4.2 ± 1.0	53.8 ± 0.2	42.3 ± 1.5	10.8 ± 1.6	14.9 ± 1.2	23.9 ± 0.9
Appeals+PGFC (Ours)	20.8 ± 0.9	3.3 ± 1.5	58.1 ± 1.3	**47.6 ± 0.6**	12.6 ± 1.7	18.5 ± 0.6	26.8 ± 1.0
Qwen2.5-Math-7B							
Base	36.2 ± 1.2	23.3 ± 1.0	81.9 ± 1.3	66.6 ± 0.5	11.8 ± 0.6	33.9 ± 0.7	42.3 ± 0.9
Rule	50.0 ± 0.1	28.7 ± 0.8	83.4 ± 0.6	82.8 ± 0.5	**29.0 ± 1.1**	45.8 ± 1.3	53.3 ± 0.7
LLM-as-Judge	41.7 ± 0.3	22.1 ± 0.3	81.9 ± 1.5	69.0 ± 0.2	13.2 ± 1.2	39.4 ± 1.1	44.6 ± 0.8
Appeals	47.1 ± 1.0	**30.4 ± 0.3**	**84.4 ± 1.2**	80.8 ± 0.3	23.5 ± 1.3	45.6 ± 0.5	52.0 ± 0.8
Appeals+PGFC (Ours)	**54.6 ± 1.4**	**30.4 ± 1.2**	82.8 ± 0.2	**83.2 ± 1.3**	**29.0 ± 0.6**	**47.6 ± 0.9**	**54.6 ± 0.9**

Table 4: Appeals statistics for Qwen2.5-Math-1.5B real-world verifier noise. Appeals+PGFC uses the same appeal stream both for recovery and for online FN-rate estimation.

Method	Avg pass@1	Avg pass@8	Appeal rate	Flip rate	TinyV calls
Rule	31.6±0.4	**42.6±1.0**	0.000	0.0000	0
Appeals	30.1±0.6	41.5±0.8	0.081	0.2288	7,963
Appeals+PGFC	**33.7±0.8**	42.4±0.5	0.093	0.2745	8,255

Table 5: Complementarity with a stronger verifier-side baseline on Qwen2.5-Math-1.5B. The correction still improves over a stronger checker, indicating that PGFC is not merely replacing verifier engineering.

Method	MATH500	AIME2024	AIME2025	AMC2023	Minerva Math	OlympiadBench	Average
Stronger checker	68.1 / 69.2	12.3 / 18.4	7.2 / 12.3	47.9 / 59.7	17.8 / 18.6	30.9 / 33.1	30.7 / 35.2
Appeals+PGFC	**72.0 / 75.1**	**14.2 / 21.6**	**8.8 / 15.8**	**51.8 / 65.8**	**20.9 / 23.8**	**33.3 / 37.2**	**33.5 / 39.9**

Table 6: General capability retention after RLVR training on *Llama-3.2-3B-Instruct*. Metrics are accuracies.

Method	MMLU	ARC-Challenge	HellaSwag	GPQA
Base	60.7	73.9	55.0	28.4
Oracle	61.1	73.9	55.9	29.1
Noise	50.0	43.9	33.9	19.4
PGBC	60.8	73.8	55.6	28.8
PGFC	**61.3**	**74.2**	**56.1**	**29.2**

C.5 General Capability Retention

We also check whether the corrections preserve non-math general benchmark performance relative to noisy training. Table 6 reports that PGBC and PGFC recover most of the retention loss caused by uncorrected noisy rewards and remain close to Base/Oracle on these auxiliary evaluations.

D Proofs and Derivations

D.1 Proof of Proposition 3.2

Proof. We compute the expectation of the noisy reward $\tilde{R}$ conditioned on the clean reward R^*, which is a binary variable. By the definition of expectation:

$$\begin{aligned}\mathbb{E}[\tilde{R}] &= 1 \cdot \mathbb{P}(\tilde{R} = 1 \mid R^*) + 0 \cdot \mathbb{P}(\tilde{R} = 0 \mid R^*) \\ &= \mathbb{P}(\tilde{R} = 1 \mid R^*).\end{aligned}$$

We can expand this using the law of total probability, conditioning on the value of $R^* \in \{0, 1\}$:

$$\mathbb{E}[\tilde{R} \mid R^*] = R^* \cdot \mathbb{P}(\tilde{R} = 1 \mid R^* = 1) + (1 - R^*) \cdot \mathbb{P}(\tilde{R} = 1 \mid R^* = 0).$$

From Definition 3.1, we have $\mathbb{P}(\tilde{R} = 1 \mid R^* = 0) = \rho_0$ and $\mathbb{P}(\tilde{R} = 0 \mid R^* = 1) = \rho_1$, which implies $\mathbb{P}(\tilde{R} = 1 \mid R^* = 1) = 1 - \rho_1$. Substituting these values:

$$\begin{aligned}\mathbb{E}[\tilde{R} \mid R^*] &= R^*(1 - \rho_1) + (1 - R^*)\rho_0 \\ &= R^* - \rho_1 R^* + \rho_0 - \rho_0 R^* \\ &= (1 - \rho_0 - \rho_1)R^* + \rho_0.\end{aligned}$$

This completes the proof. □

D.2 Proof of Theorem 3.3

Proof. Recall the verifier reward channel (Definition 3.1): for the latent clean reward $R^* \in \{0, 1\}$ and the observed noisy reward $\tilde{R} \in \{0, 1\}$,

$$\Pr(\tilde{R} = 1 \mid R^* = 0) = \rho_0, \qquad \Pr(\tilde{R} = 0 \mid R^* = 1) = \rho_1,$$

hence $\Pr(\tilde{R} = 1 \mid R^* = 1) = 1 - \rho_1$ and $\Pr(\tilde{R} = 0 \mid R^* = 0) = 1 - \rho_0$. Since $1 - \rho_0 - \rho_1 > 0$, the estimator

$$\widehat{R} \triangleq \frac{\tilde{R} - \rho_0}{1 - \rho_0 - \rho_1}$$

is well-defined.

We prove unbiasedness by computing the conditional expectation given R^*. First note that because $\tilde{R}$ is binary,

$$\mathbb{E}[\tilde{R} \mid R^*] = 1 \cdot \Pr(\tilde{R} = 1 \mid R^*) + 0 \cdot \Pr(\tilde{R} = 0 \mid R^*) = \Pr(\tilde{R} = 1 \mid R^*). \tag{7}$$

By the channel definition, for the two possible values of R^* we have

$$\mathbb{E}[\tilde{R} \mid R^* = 1] = 1 - \rho_1, \qquad \mathbb{E}[\tilde{R} \mid R^* = 0] = \rho_0.$$

These two cases can be written compactly as the affine form (Proposition 3.2)

$$\mathbb{E}[\tilde{R} \mid R^*] = (1 - \rho_0 - \rho_1)R^* + \rho_0. \tag{8}$$

Now take the conditional expectation of $\widehat{R}$:

$$\begin{aligned}
\mathbb{E}[\widehat{R} \mid R^*] &= \mathbb{E}\left[\frac{\tilde{R} - \rho_0}{1 - \rho_0 - \rho_1} \middle| R^* \right] \\
&= \frac{\mathbb{E}[\tilde{R} \mid R^*] - \rho_0}{1 - \rho_0 - \rho_1} \\
&= \frac{(1 - \rho_0 - \rho_1)R^* + \rho_0 - \rho_0}{1 - \rho_0 - \rho_1} \quad \text{(by (8))} \\
&= R^*. \qquad (9)
\end{aligned}$$

Equation (9) shows that $\widehat{R}$ is an unbiased estimator of the latent clean reward in the conditional sense.

Finally, taking total expectation (law of iterated expectation) yields

$$\mathbb{E}[\widehat{R}] = \mathbb{E}\big[\mathbb{E}[\widehat{R} \mid R^*]\big] = \mathbb{E}[R^*],$$

so $\widehat{R}$ is unbiased for the clean reward both conditionally and unconditionally. □

D.3 Proof of Proposition 3.4

Proof. The proposition states two claims about the conditional expectation of the forward weights. The weights are defined as:

$$w_{\tilde{R}} = \begin{cases} w_0 = \rho_1 - 1 & \text{if } \tilde{R} = 0, \\ w_1 = \rho_1 & \text{if } \tilde{R} = 1. \end{cases}$$

The noise model provides the conditional probabilities:

$$\begin{aligned}
&\Pr(\tilde{R} = 0 \mid R^* = 1) = \rho_1, \qquad \Pr(\tilde{R} = 1 \mid R^* = 1) = 1 - \rho_1 \\
&\Pr(\tilde{R} = 1 \mid R^* = 0) = \rho_0, \qquad \Pr(\tilde{R} = 0 \mid R^* = 0) = 1 - \rho_0
\end{aligned}$$

Part 1: Proof of $\mathbb{E}[w_{\tilde{R}} \mid R^* = 1] = 0$ We compute the expectation of $w_{\tilde{R}}$ conditioned on the true reward being positive ($R^* = 1$):

$$\begin{aligned}
\mathbb{E}[w_{\tilde{R}} \mid R^* = 1] &= \sum_{k \in \{0,1\}} w_k \cdot \Pr(\tilde{R} = k \mid R^* = 1) \\
&= w_0 \cdot \Pr(\tilde{R} = 0 \mid R^* = 1) + w_1 \cdot \Pr(\tilde{R} = 1 \mid R^* = 1) \\
&= (\rho_1 - 1) \cdot (\rho_1) + (\rho_1) \cdot (1 - \rho_1) \\
&= (\rho_1^2 - \rho_1) + (\rho_1 - \rho_1^2) \\
&= 0.
\end{aligned}$$

Part 2: Proof of $\mathbb{E}[w_{\tilde{R}} \mid R^* = 0] = -(1 - \rho_0 - \rho_1)$ Next, we compute the expectation of $w_{\tilde{R}}$ conditioned on the true reward being negative ($R^* = 0$):

$$\begin{aligned}
\mathbb{E}[w_{\tilde{R}} \mid R^* = 0] &= \sum_{k \in \{0,1\}} w_k \cdot \Pr(\tilde{R} = k \mid R^* = 0) \\
&= w_0 \cdot \Pr(\tilde{R} = 0 \mid R^* = 0) + w_1 \cdot \Pr(\tilde{R} = 1 \mid R^* = 0) \\
&= (\rho_1 - 1) \cdot (1 - \rho_0) + (\rho_1) \cdot (\rho_0) \\
&= (\rho_1 - \rho_0\rho_1 - 1 + \rho_0) + \rho_0\rho_1 \\
&= \rho_1 + \rho_0 - 1 \\
&= -(1 - \rho_0 - \rho_1).
\end{aligned}$$

This proves both claims of the proposition. □

D.4 Proof of Theorem 3.5

Proof. We want to show that $\mathbb{E}[\Delta\theta] = (1 - \rho_0 - \rho_1)\nabla_\theta J(\theta)$, where $\Delta\theta = \frac{1}{M}\sum_{t=1}^{M} h_t$ and $h_t = w_{\tilde{R}} G_t$. By linearity of expectation and assuming i.i.d. samples, it suffices to show this for a single sample's contribution, $\mathbb{E}[h_t]$.

We use the law of total expectation, conditioning on the latent true reward $R^* \in \{0, 1\}$:

$$\begin{aligned}\mathbb{E}[h_t] = \mathbb{E}[w_{\tilde{R}} G_t] &= \mathbb{E}\left[\mathbb{E}[w_{\tilde{R}} G_t \mid R^*]\right] \\ &= \Pr(R^* = 1)\,\mathbb{E}[w_{\tilde{R}} G_t \mid R^* = 1] + \Pr(R^* = 0)\,\mathbb{E}[w_{\tilde{R}} G_t \mid R^* = 0].\end{aligned}$$

The noise process generating $\tilde{R}$ is independent of the policy's action generation process (which produces G_t), conditional on the true reward R^*. Thus, we can separate the expectations:

$$\mathbb{E}[w_{\tilde{R}} G_t \mid R^*] = \mathbb{E}[w_{\tilde{R}} \mid R^*] \cdot \mathbb{E}[G_t \mid R^*].$$

Using the results from Proposition 3.4:

- $\mathbb{E}[w_{\tilde{R}} \mid R^* = 1] = 0$.
- $\mathbb{E}[w_{\tilde{R}} \mid R^* = 0] = -(1 - \rho_0 - \rho_1)$.

Substituting these back into the main expectation formula:

$$\begin{aligned}\mathbb{E}[w_{\tilde{R}} G_t] &= \Pr(R^* = 1) \cdot (0) \cdot \mathbb{E}[G_t \mid R^* = 1] \\ &\quad + \Pr(R^* = 0) \cdot (-(1 - \rho_0 - \rho_1)) \cdot \mathbb{E}[G_t \mid R^* = 0] \\ &= -(1 - \rho_0 - \rho_1) \cdot \Pr(R^* = 0)\mathbb{E}[G_t \mid R^* = 0] \\ &= -(1 - \rho_0 - \rho_1) \cdot \mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t],\end{aligned}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function. From two fundamental properties of the score function:

1. The unconditional expectation is zero: $\mathbb{E}[G_t] = 0$ (Williams, 1992; Sutton et al., 1999).
2. The clean policy gradient is $\nabla_\theta J(\theta) = \mathbb{E}[R^* G_t]$.

From property 1, we have $\mathbb{E}[G_t] = \mathbb{E}[(\mathbf{1}_{\{R^*=1\}} + \mathbf{1}_{\{R^*=0\}})G_t] = \mathbb{E}[R^* G_t] + \mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t] = 0$. This implies that $\mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t] = -\mathbb{E}[R^* G_t] = -\nabla_\theta J(\theta)$.

Finally, we substitute this back into our expression for the expected update direction:

$$\begin{aligned}\mathbb{E}[h_t] &= \mathbb{E}[w_{\tilde{R}} G_t] \\ &= -(1 - \rho_0 - \rho_1) \cdot \mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t] \\ &= -(1 - \rho_0 - \rho_1) \cdot (-\nabla_\theta J(\theta)) \\ &= (1 - \rho_0 - \rho_1)\nabla_\theta J(\theta).\end{aligned}$$

Therefore, the expectation of the full update is $\mathbb{E}[\Delta\theta] = \frac{1}{M}\sum \mathbb{E}[h_t] = (1 - \rho_0 - \rho_1)\nabla_\theta J(\theta)$. This completes the proof. □

E Prompt Templates and Training/Evaluation Details

This section records the exact prompt formats and the concrete hyperparameters we used for all experiments in this paper.

E.1 Prompt Templates

Training (generation) prompt. For each math problem x (a plain-text question), the user message is built by concatenating the raw question with a short instruction that elicits chain-of-thought and enforces a verifiable answer format.

```
<user>
{QUESTION}

Let's think step by step and enclose the reasoning process within <
    think> and </think> tags.
The final result in the answer MUST BE within \boxed{}.
</user>
```

During data preprocessing, we write chat-style JSON with a single user turn as shown above and attach the rule-based ground-truth answer for reward checking.

Evaluation (validation/test) prompt. We use the same prompt template as training for validation and test-time generation so that the rule-based verifier can parse the boxed answer consistently.

Verifier I/O. The **rule-based** checker operates on the model's final string and extracts the last `\boxed{...}` expression; it then applies numeric/rational parsing and equality tests to produce a binary reward $\tilde{R} \in \{0, 1\}$. When the **LLM verifier** is enabled, it receives the pair $(\text{problem}, \text{model solution})$ and returns a binary correctness decision used only to estimate the false negative rate ρ_1 over a sliding window. The LLM verifier does not replace the rule-based reward.

E.2 Data Preprocessing

We load the preview split of the math-reasoning corpus and map each example to a chat-style record as above, keeping the reference (ground-truth) answer for programmatic checking.

E.3 Training Configuration

Table 7: Core training settings.

Training (GRPO)	
Train batch size	128
Rollouts per question (group size)	8
Max prompt length (tokens)	512
Max response length (tokens)	3072
Sampling temperature (rollouts)	1.0
Advantage estimator	GRPO
KL regularization	Enabled
KL coefficient β	0.001
Entropy coefficient	0.0
Optimizer	AdamW
Learning rate	3e-6
Total epochs	1

Model/backbone. We load the base model from local cache (HuggingFace layout), enable FSDP2 for actor/ref, and use shared-memory weights with remove-padding for efficient vLLM rollouts. The KL is computed w.r.t. a frozen reference initialized from the same base.

E.4 Online FN-Rate Estimation Hyperparameters and Ablations

This subsection reports the concrete hyperparameters for our online false-negative (FN) rate estimation $\hat{\rho}_1$ used in the real-world-noise experiments (Section 4.3).

Appeals protocol and estimator. At training step t, let $\mathcal{N}_{\mathrm{R}}^{(t)}$ and $\mathcal{P}_{\mathrm{R}}^{(t)}$ be the sets labeled negative/positive by the primary rule-based verifier. We appeal each item in $\mathcal{N}_{\mathrm{R}}^{(t)}$ independently with probability $q \in (0, 1]$ and re-evaluate it with a lightweight LLM verifier (TinyV). Let $\mathcal{P}_{\mathrm{L}}^{(t)}$ be the

appealed negatives that TinyV flips to positive. We use the Horvitz–Thompson estimator

$$\widehat{\mathrm{FN}}^{(t)} \;=\; \frac{|\mathcal{P}_{\mathrm{L}}^{(t)}|}{q} \tag{10}$$

and estimate the FN rate with a Beta prior:

$$\hat{\rho}_1^{(t)} \;=\; \frac{\widehat{\mathrm{FN}}^{(t)} + \alpha}{\widehat{\mathrm{FN}}^{(t)} + |\mathcal{P}_{\mathrm{R}}^{(t)}| + \alpha + \beta}\,. \tag{11}$$

We then apply exponential moving average (EMA) smoothing

$$\bar{\rho}_1^{(t)} \;=\; \lambda\,\bar{\rho}_1^{(t-1)} + (1-\lambda)\,\hat{\rho}_1^{(t)}\,, \tag{12}$$

and use $\bar{\rho}_1^{(t)}$ as the plug-in estimate for PGFC weights $w_{\tilde{r}} \in \{\bar{\rho}_1^{(t)} - 1,\ \bar{\rho}_1^{(t)}\}$.

Default hyperparameters. Unless otherwise stated, real-world-noise runs use: (i) appeal probability $q = 0.1$; (ii) Beta prior pseudocounts $\alpha = \beta = 10^{-5}$ in Eq. (11); (iii) EMA decay $\lambda = 0.9$ in Eq. (12). We compute $\hat{\rho}_1^{(t)}$ and update $\bar{\rho}_1^{(t)}$ once per policy-update step. The expected number of TinyV calls per update is $q \cdot |\mathcal{N}_{\mathrm{R}}^{(t)}|$.

E.5 Datasets

Training.

- **DeepScaleR** (Luo et al., 2025): the math-reasoning corpus used for RLVR rollouts and policy updates.

Evaluation. We evaluate with the standard official/problem releases for each benchmark.

- **MATH500** (Lightman et al., 2024): a verifiable subset of math reasoning problems for reliable evaluation.
- **AIME 2024** (HuggingFaceH4, 2024): the 2024 American Invitational Mathematics Examination set.
- **AIME 2025** (OpenCompass, 2025): the 2025 AIME set.
- **AMC 2023** (math-ai, 2025): problems from the 2023 American Mathematics Competitions.
- **Minerva Math** (Lewkowycz et al., 2022): the Minerva math evaluation suite.
- **OlympiadBench** (He et al., 2024): olympiad-level mathematics benchmark.
- **MMLU** (Hendrycks et al., 2021): a broad multi-task benchmark covering 57 academic and professional subjects for evaluating world knowledge and problem solving.
- **ARC-C** (Clark et al., 2018): the Challenge split of the AI2 Reasoning Challenge, consisting of grade-school science questions that are difficult for retrieval and word co-occurrence baselines.
- **HellaSwag** (Zellers et al., 2019): a commonsense natural language inference benchmark for physically grounded sentence-completion reasoning.
- **GPQA** (Rein et al., 2024): a graduate-level, Google-proof multiple-choice QA benchmark written by domain experts in biology, physics, and chemistry.

F False Positive (FP) and False Negative (FN) Examples

We also provide typical failure modes observed during math RL with automated verifiers on `Qwen2.5-Math-7B` under `MATH500` dataset. For rule-based verifier: 52 FNs and 0 FP out of 500 items (marginal rates 10.4% and 0%); for LLM verifier: 81 FNs and 168 FPs out of 500 items (marginal rates 16.2% and 33.6%). The FP case shows an LLM judge over-crediting a malformed solution, while the FN case shows a rule-based checker rejecting a numerically correct answer due to formatting. The raw JSON context for these examples is provided below.

False Positive (FP).
Problem. Compute: $1-2+3-4+\cdots+99-100$.
Ground truth. $\boxed{-50}$
Model response (excerpt).

```
Compute: $1+2-3-4+5+6-7-8+\dots+97+98-99-100$.
\boxed{ } \boxed{ } \boxed{ } \dots
```

Verifier signals. Rule-based checker: ✗ (incorrect); LLM judge: ✓ (incorrectly accepts) ⇒ **FP**.

False Negative (FN).
Problem. Each of seven islands independently has treasure with probability $\frac{1}{5}$. What is the probability that exactly four islands have treasure?
Correct answer. $\boxed{\frac{448}{15625}}$ (equals 0.028672).
Model answer. $\boxed{0.028672}$
Verifier signals. Rule-based exact-match: ✗ (expects a fraction); numeric equivalence check: ✓ (values match) ⇒ **FN**.

We provide FP and FN examples using Qwen2.5-Math-7B base model on MATH500.

FP


```
{
    "idx": 16,
    "question": "Compute: $1-2+3-4+5- \\dots +99-100$.",
    "model_prediction": "{}",
    "model_response": "Compute: $1+2-3-4+5+6-7-8+ \\dots +97+98-99-100$.
      \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}.
    \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}.  \\",
    "ground_truth": "-50",
    "rule_based": false,
    "llm_verifier": true,
    "llm_verifier_score": 1.0,
    "llm_verifier_raw_value": "1.0",
    "math_verify": false,
    "math_verify_score": 0.0
  },
```


FN


```
{
    "idx": 92,
    "question": "A pirate searches seven islands for buried treasure.
    If each island has a $\\frac{1}{5}$ chance of having treasure, what
    is the probability that exactly 4 of the islands have treasure?",
    "answer": "\\frac{448}{15625}",
    "model_prediction": "0.028672",
```

```
    "model_response": "To solve this problem, we can use the binomial
    probability formula. The probability of having exactly $k$ successes
     in $n$ independent Bernoulli trials (each with success probability
    $p$) is given by:\n\n\\[ P(X = k) = \\binom{n}{k} p^k (1-p)^{n-k}
    \\]\n\nHere, $n = 7$ (the number of islands), $k = 4$ (the number of
     islands with treasure), and $p = \\frac{1}{5}$ (the probability
    that a single island has treasure).\n\nThe binomial coefficient $\\
    binom{n}{k}$ is the number of ways to choose $k$ successes out of
    $n$ trials, and it is calculated as:\n\n\\[ \\binom{n}{k} = \\frac{n
    !}{k!(n-k)!} \\]\n\n So, the probability that exactly 4 of the 7
    islands have treasure is:\n\n\\[ P(X = 4) = \\binom{7}{4} \\left(\\
    frac{1}{5}\\right)^4 \\left(\\frac{4}{5}\\right)^{3} \\]\n\nLet's
    calculate this using Python.\n```python\nimport math\n\n# Number of
    islands\nn = 7\n# Number of islands with treasure\nk = 4\n#
    Probability of a single island having treasure\np = 1/5\n\n#
    Binomial coefficient\nbinom_coeff = math.comb(n, k)\n\n# Probability
     calculation\nprobability = binom_coeff * (p**k) * ((1-p)**(n-k))\
    nprint(probability)\n```\n```output\n0.028672000000000001\n```\nThe
    probability that exactly 4 of the 7 islands have treasure is \\(\\
    boxed{0.028672}\\).",
    "ground_truth": "\\frac{448}{15625}",
    "rule_based": false,
    "llm_verifier": false,
    "llm_verifier_score": 0.0,
    "llm_verifier_raw_value": "0.0",
    "math_verify": true,
    "math_verify_score": 1.0
  }
```


G Additional Evaluations and Analysis

G.1 Estimated-Rate Corrections with CompassVerifier

TinyV appeals a random subset of rule-negative responses and supports false-negative-rate estimation under the FN-dominant assumption. This stream does not independently audit rule-positive outputs and therefore does not supply a calibrated estimate of ρ_0. PGBC additionally requires that rate. We evaluate Qwen2.5-Math-1.5B with CompassVerifier-3B (Liu et al., 2025) using stratified audits of both rule-negative and rule-positive outputs. PGFC uses the negative-side estimate of ρ_1, while PGBC uses estimates of both rates. Both corrections improve over direct verifier training and Compass Appeals in Table 8.

Table 8: Qwen2.5-Math-1.5B with CompassVerifier-3B and estimated channel rates. Scores are percentages.

Method	AIME 2024	AIME 2025	AMC 2023	MATH-500	Minerva MATH	OlympiadBench
Rule	15.0 ± 0.4	5.6 ± 0.6	50.3 ± 0.6	69.4 ± 0.4	17.8 ± 0.6	31.6 ± 0.0
LLM-Verifier	16.4 ± 1.0	7.2 ± 0.9	51.5 ± 1.0	70.8 ± 0.7	18.5 ± 0.8	32.0 ± 0.6
Compass Appeals	16.7 ± 0.8	7.4 ± 0.8	51.8 ± 0.8	71.0 ± 0.6	18.9 ± 0.7	32.3 ± 0.5
Compass Appeals+PGBC	18.7 ± 0.7	8.9 ± 0.8	53.0 ± 0.7	71.7 ± 0.5	21.5 ± 0.6	33.1 ± 0.5
Compass Appeals+PGFC	19.8 ± 0.4	9.8 ± 0.1	54.0 ± 0.3	72.1 ± 0.7	20.1 ± 0.3	33.8 ± 0.9

G.2 Varying the True Noise Channel

The rate-misspecification experiment varies supplied estimates around a fixed true channel. Table 9 instead changes the true input channel and reports average accuracy across the six math datasets. PGBC performance decreases as $c = 1 - \rho_0 - \rho_1$ shrinks, while PGFC is more stable over the tested range. This pattern is consistent with the inverse factor in PGBC; it does not imply that PGFC can recover a signal from an uninformative channel.

Table 9: True-channel sweep: average accuracy (%) across six math benchmarks.

ρ_0	ρ_1	c	Noise	PGBC	PGFC	Oracle
0.05	0.10	0.85	27.0 ± 0.8	31.5 ± 0.6	31.1 ± 0.3	31.8 ± 0.5
0.10	0.20	0.70	20.4 ± 0.8	31.1 ± 0.6	31.6 ± 0.6	31.8 ± 0.5
0.15	0.30	0.55	17.2 ± 1.0	29.4 ± 0.8	30.7 ± 0.7	31.8 ± 0.5
0.20	0.40	0.40	14.5 ± 1.2	26.4 ± 1.0	30.5 ± 0.9	31.8 ± 0.5

G.3 Auxiliary-Verifier Calibration and Estimation Error

Let s and f be the auxiliary verifier's sensitivity and false-positive rate on appealed rule negatives, and let π denote the prevalence of true false negatives in that pool. Provided $s \neq f$, calibration gives

$$p_{\mathrm{flip}} = f + (s - f)\pi, \qquad \widehat{\pi} = \mathrm{clip}_{[0,1]}\left(\frac{\widehat{p}_{\mathrm{flip}} - \widehat{f}}{\widehat{s} - \widehat{f}}\right), \tag{13}$$

$$\widehat{\rho}_1 = \frac{|\mathcal{N}_{\mathrm{R}}|\widehat{\pi}}{|\mathcal{N}_{\mathrm{R}}|\widehat{\pi} + |\mathcal{P}_{\mathrm{R}}|}. \tag{14}$$

The final expression assumes negligible false positives from the rule verifier. The uncalibrated estimator corresponds to $s = 1$ and $f = 0$ before smoothing and pseudocounts. Deployment-time calibration of TinyV on appealed rule negatives yields Table 10.

Table 10: Reported deployment-time calibration on appealed rule-negative responses.

Statistic	Value
Sensitivity on true false negatives	0.93
False-positive rate on true negatives	0.03
Naive $\widehat{\rho}_1$	0.785
Calibration-corrected $\widehat{\rho}_1$	0.780

For an estimate based on historical data and independent of the current rollout, condition on its error $\delta = \widehat{\rho}_1 - \rho_1$. Both PGFC weights shift by the same constant: $\widehat{w}_{\widetilde{R}} = w_{\widetilde{R}} + \delta$. With $G = \nabla_\theta \log \pi_\theta(y \mid x)$, the score identity $\mathbb{E}[G] = 0$ implies

$$\mathbb{E}[\widehat{w}_{\widetilde{R}} G] = \mathbb{E}[w_{\widetilde{R}} G], \tag{15}$$

$$\mathbb{E}\big[\|\widehat{g}(\widehat{\rho}_1) - \widehat{g}(\rho_1)\|^2\big] = \frac{\delta^2}{M}\,\mathrm{tr}(\mathrm{Cov}(G)) \tag{16}$$

for the average of M independent rollouts. Thus this ideal unclipped estimator has no population-gradient bias from a fixed rate error, and the mean-square finite-sample perturbation is quadratic in δ. Same-batch estimation and clipping can introduce residual effects; this identity is not a guarantee for the full clipped training objective.

G.4 Conditional Advantage and Gradient Variance

We pool final trajectory-level advantages across rollout groups conditional on clean correctness. For method m, define $\mathcal{I}_r = \{(b,i) : R^*_{bi} = r\}$, $n_r = |\mathcal{I}_r|$, and $\bar{a}_r^{(m)} = n_r^{-1} \sum_{(b,i)\in\mathcal{I}_r} a_{bi}^{(m)}$. The conditional sample variance is

$$\widehat{\mathrm{Var}}(a^{(m)} \mid R^* = r) = \frac{1}{n_r - 1} \sum_{(b,i)\in\mathcal{I}_r} \left(a_{bi}^{(m)} - \bar{a}_r^{(m)}\right)^2, \qquad r \in \{0,1\}. \tag{17}$$

Table 11 also reports gradient covariance trace relative to Noise and the coefficient of variation (CV) of gradient norms. PGBC has larger conditional advantage variance in both classes and a larger covariance trace than either Noise or PGFC in this measurement.

Table 11: Direct variance measurements. Gradient covariance trace is normalized to the Noise value.

Method	$\widehat{\mathrm{Var}}(a \mid R^* = 0)$	$\widehat{\mathrm{Var}}(a \mid R^* = 1)$	$\mathrm{tr}\,\widehat{\mathrm{Cov}}(g)$	Gradient-norm CV
Noise	0.107	0.482	$1.000\times$	0.729
PGBC	0.671	1.194	$3.046\times$	0.873
PGFC	0.181	0.382	$0.905\times$	0.730

G.5 Direct-Coefficient REINFORCE

To examine the corrections outside the clipped GRPO variant, we additionally optimize a direct-coefficient REINFORCE estimator. For each response, set

$$u_i = \begin{cases} R_i^*, & \text{Oracle}, \\ \widetilde{R}_i, & \text{Noise}, \\ (\widetilde{R}_i - \widehat{\rho}_0)/(1 - \widehat{\rho}_0 - \widehat{\rho}_1), & \text{PGBC}, \\ \widetilde{R}_i - (1 - \widehat{\rho}_1), & \text{PGFC}. \end{cases} \tag{18}$$

Broadcasting $A_{i,t} = u_i$ to response tokens yields

$$\widehat{g} = \frac{1}{K} \sum_{i=1}^{K} u_i \sum_t \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t}). \tag{19}$$

Both corrections improve over Noise on every benchmark in Table 12. This comparison provides evidence for the correction beyond the clipped surrogate used in the main experiments.

Table 12: Additional direct-coefficient REINFORCE results on Qwen2.5-Math-1.5B.

Method	AIME 2024	AIME 2025	AMC 2023	MATH-500	Minerva MATH	OlympiadBench
Oracle	10.8 ± 0.2	7.6 ± 0.1	50.9 ± 0.4	69.3 ± 0.1	29.2 ± 0.5	33.4 ± 0.3
Noise	7.1 ± 0.7	4.2 ± 0.6	37.8 ± 0.8	54.6 ± 0.5	15.8 ± 1.0	25.9 ± 0.6
PGBC	10.2 ± 0.5	9.1 ± 0.4	47.8 ± 0.7	67.9 ± 0.4	30.6 ± 0.8	31.4 ± 0.5
PGFC	11.1 ± 0.4	9.4 ± 0.3	48.4 ± 0.5	69.2 ± 0.3	30.0 ± 0.7	32.2 ± 0.4

G.6 Increasing the Rollout Count

Table 13 compares Qwen2.5-Math-1.5B with $K \in \{8, 16, 32\}$ rollouts. Increasing the sample count improves Noise, but a gap to Oracle remains at every tested K. More rollouts reduce Monte Carlo variance; they do not remove the underlying reward-channel distortion.

Table 13: Additional rollout-count comparison on Qwen2.5-Math-1.5B.

Rollouts K	Oracle	Noise	PGBC	PGFC
8	33.5 ± 0.3	24.2 ± 0.7	32.8 ± 0.6	33.4 ± 0.5
16	34.0 ± 0.5	26.0 ± 0.8	33.2 ± 0.4	33.8 ± 0.2
32	35.2 ± 0.3	28.7 ± 0.4	34.4 ± 0.5	35.0 ± 0.4

H Additional Scope Clarifications

H.1 Informative, Uninformative, and Reversed Channels

The assumption $\rho_0 + \rho_1 < 1$ means that correct responses are more likely to receive positive reward than incorrect responses:

$$\Pr(\widetilde{R} = 1 \mid R^* = 1) = 1 - \rho_1 > \rho_0 = \Pr(\widetilde{R} = 1 \mid R^* = 0). \tag{20}$$

At $\rho_0 + \rho_1 = 1$, the two conditional reward distributions coincide and the transition matrix is singular; observed rewards alone carry no correctness information under this channel model. If $\rho_0 + \rho_1 > 1$, the channel is anti-informative. When a calibration set identifies that orientation, complementing $\widetilde{R}' = 1 - \widetilde{R}$ gives $\rho_0' = 1 - \rho_0$ and $\rho_1' = 1 - \rho_1$, so $\rho_0' + \rho_1' = 2 - (\rho_0 + \rho_1) < 1$. The correction can then use the complemented channel. As $c = 1 - \rho_0 - \rho_1$ approaches zero from above, PGBC amplifies coefficients through $1/c$, whereas PGFC's expected signal shrinks proportionally to c.

For content-dependent errors $\rho_0(x, y)$ and $\rho_1(x, y)$, constant-rate corrections operate on an effective channel. Dependence between verifier errors and score terms can leave a residual such as $\mathrm{Cov}(w_{\widetilde{R}}, G \mid R^*)$. The non-i.i.d. experiment tests empirical behavior under this misspecification; it does not extend the exact population theorem to arbitrary content-dependent noise.

H.2 Single-Reward Functions and Nonlinear Group Objectives

Every function of one binary reward is affine:

$$f(R^*) = f(0) + \big(f(1) - f(0)\big)R^*. \tag{21}$$

Replacing R^* by its backward-corrected estimator therefore provides an unbiased estimate of $f(R^*)$ under the stated channel assumptions. For a function $F(R_1^*, \ldots, R_K^*)$ of an entire group with independent verifier noise, the channel is $T^{\otimes K}$. Inverting this channel can in principle produce an unbiased estimator when T is invertible, but direct inversion becomes poorly conditioned as group size increases for a noisy channel.

For example, the clean pass@K outcome is $1 - \prod_{j=1}^{K}(1 - R_j^*)$. With independent verifier noise and correctly specified rates, an unbiased estimator is

$$1 - \prod_{j=1}^{K}(1 - \widehat{R}_j), \tag{22}$$

where $\widehat{R}_j$ is the backward-corrected reward. A policy gradient for this group objective includes score terms from all K sampled responses. For a nonlinear function $g(p_x)$ of prompt success probability,

$$p_x = \frac{\mathbb{E}[\widetilde{R} \mid x] - \rho_0}{1 - \rho_0 - \rho_1}, \tag{23}$$

but the plug-in estimate $g(\widehat{p}_x)$ is generally not finite-sample unbiased. Clipping or shrinking $\widehat{p}_x$ and using uncertainty-aware weights may be useful practical extensions, but these require separate analysis and experimental validation.