TRACEBAND: PREFIX REWEIGHTING UNDER FINITE-SAMPLE UNCERTAINTY FOR RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

In reinforcement learning with verifiable rewards (RLVR), sampled continuations can be used to estimate the values of intermediate reasoning prefixes, but finite sampling leaves these estimates uncertain. This uncertainty makes it unclear whether observed differences reflect true value differences and therefore justify reweighting the sampled prefixes. We show that directly reweighting sampled prefixes using these finite-sample estimates can decrease the sampled-prefix objective. We introduce *TraceBand*, which constructs simultaneous confidence bands for prefix values and reweights sampled prefixes to maximize worst-case objective improvement. The resulting maximin problem has a closed-form solution that leaves the sampled-prefix distribution unchanged whenever the intervals have a nonempty intersection, that is, whenever all prefixes may have the same value, in which case reweighting could only incur a cost without any gain. Whenever the intervals contain the true prefix values, this reweighting has nonnegative gain in the sampled-prefix objective relative to the unchanged prefix weights. Experiments across mathematical reasoning and code generation show that TraceBand improves over finite-sample plug-in reweighting and achieves strong performance relative to representative RLVR baselines across multiple models.

1 INTRODUCTION

Many mathematics and programming tasks admit programmatically checkable outcomes, such as exact-answer verification or execution against test cases. Reinforcement learning with verifiable rewards (RLVR) exploits this structure to train language models from rewards assigned to completed responses (Guo et al., 2025; Liu et al., 2025a; Stojanovski et al., 2025; Zeng et al., 2025). Because the RL reward can be constructed from the final response, this training stage does not require annotations for every intermediate reasoning step.

Response-level methods such as GRPO provide only coarse credit because a completion-level reward does not distinguish the intermediate prefixes that produced the response (Shao et al., 2024; Guo et al., 2025). To provide finer-grained credit, prefix-level methods estimate intermediate values from multiple descendant continuations, either by explicitly branching from intermediate states or by reusing continuation structure already present in sampled responses (Kazemnejad et al., 2025; Wang et al., 2026; Sun et al., 2026; Tran et al., 2026). Recent methods such as GRPO-MA and PPPO use multiple continuation outcomes to reduce sampling noise in prefix-level credit, but they do not eliminate finite-sample uncertainty (Wang et al., 2026; Sun et al., 2026).

The remaining uncertainty can matter even after multiple continuations have been sampled. Consider two prefixes that produce five and four successful continuations, respectively, out of eight trials. A point-estimate update assigns a more favorable training signal to the first prefix, even though their binomial confidence intervals overlap substantially and their ordering remains poorly resolved. Although additional continuations could reduce this ambiguity, under a fixed continuation budget the learner must decide whether, and by how much, the observed difference justifies changing the generation probabilities of the two prefixes.

We study this decision through a KL-regularized objective. Its optimal update multiplies the generation probability of each prefix by a factor that grows with the value of its continuations, so that probability moves toward higher-value prefixes (Peters et al., 2010; Rafailov et al., 2023; Xiong

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.

et al., 2024; Shu et al., 2026). We refer to this step as *prefix reweighting*. With finite samples, however, the plug-in update, which applies the same rule to estimated values, can follow estimation error and decrease the regularized objective when this error outweighs the available gain.

We introduce *TraceBand* to account for finite-sample uncertainty in prefix reweighting. Using the same continuation samples, it constructs simultaneous confidence bands for the prefix values and selects weights whose gain over the original generation probabilities is nonnegative for all values consistent with those intervals. The resulting rule has a closed form and leaves the generation probabilities unchanged when the intervals do not support reweighting. Whenever the intervals jointly cover the true values of all sampled prefixes, the resulting target preserves or improves the marginal objective on the sampled prefix pool.

Our main contributions are as follows.

- We formulate finite-sample prefix reweighting as worst-case improvement over simultaneous confidence bands and derive *TraceBand*, a closed-form reweighting rule with an explicit no-change region.
- We show that TRACEBAND maximizes worst-case improvement over feasible reweightings of the prefixes. When the confidence bands simultaneously cover the true prefix values, the resulting reweighting has nonnegative gain in the marginal objective on the fixed sampled-prefix pool.
- Across mathematical reasoning and code generation, we validate the target and its realization by the fitted policy, observe gains over alternative reweighting schemes, and achieve strong performance relative to representative RLVR baselines.

2 RELATED WORK

Prefix credit and finite-budget sampling. In RLVR the verifier scores the completed response, and critic-free algorithms such as GRPO broadcast this response-level signal to every token of the response (Shao et al., 2024). Finer credit for intermediate prefixes is obtained either from a learned model, which may be a value critic or a process reward model trained on step-level annotations or outcome supervision (Lightman et al., 2024; Wang et al., 2024; Yuan et al., 2025), or directly from sampled continuations. VinePPO estimates prefix values by Monte Carlo continuation sampling, GRPO-MA branches several continuations from a shared thought prefix and analyzes how the numbers of prefixes and continuations affect the variance of the prefix-level advantage, PPPO aggregates continuation rewards into a prefix reward, and TEMPO reuses the tree structure of sampled responses (Kazemnejad et al., 2025; Wang et al., 2026; Sun et al., 2026; Tran et al., 2026). A related line uses uncertainty to decide where computation is spent. PROS reuses historical prefixes and prioritizes prompts whose estimated success rate is near one half, and PS-PPO samples prefix cutoffs by a prefix uncertainty signal (Huang & Wan, 2026; Hwang & Kim, 2026). These methods reduce the variance of prefix-value estimates or distribute the sampling budget. TraceBand addresses the decision that follows, whether and by how much the estimated values justify changing the prefix probabilities under a fixed continuation budget.

KL-regularized policy targets. The KL-regularized objective of RL has a closed-form optimum in which the rollout policy is reweighted by the exponentiated reward. This structure underlies relative-entropy policy search (Peters et al., 2010) and the derivation of direct preference optimization and its iterative variants (Rafailov et al., 2023; Xiong et al., 2024), and MPO constructs such a target from estimated action values and then fits a parametric policy to it (Abdolmaleki et al., 2018). For RLVR, BOLT fits the KL-regularized target by weighting sampled responses with their exponentiated rewards and analyzes the finite-sample error of this fit (Shu et al., 2026). In their sample-based forms, these targets treat the estimated values as exact. TraceBand instead accounts for the uncertainty of the estimated prefix values in the target itself and reweights prefixes only as far as the samples support.

Uncertainty and baseline-relative policy improvement. Safe policy improvement changes a baseline policy only where the data support the change. High-confidence policy improvement returns the baseline unless a confidence bound on the return of the candidate exceeds it (Thomas et al.,

2015). Robust baseline regret maximizes the worst-case improvement over the baseline across a set of plausible models, evaluating both policies under the same model (Ghavamzadeh et al., 2016). SPIBB and Soft-SPIBB instead constrain the policy toward the behavior policy on state-action pairs with little data (Laroche et al., 2019; Nadjahi et al., 2019). These methods decide about a whole policy from a fixed batch, and none of them yields the update in closed form. TraceBand poses the same question for the probabilities of a few prefixes under exact confidence intervals on their values, where the maximin update has a closed-form solution. Uncertainty has also been used in RLVR to encourage exploration, as in UCAS, which shapes advantages by the model's own confidence (Xie et al., 2026). TraceBand uses the uncertainty of verifier-derived estimates to restrain updates instead.

3 TRACEBAND

A terminal reward evaluates one completed response and carries no information about the other outcomes that were possible after its prefix. Writing a response as a prefix followed by a continuation separates a training target into two components, the conditional distribution over the continuations of each prefix and the marginal distribution. TraceBand concerns the prefix marginal and makes its update depend on the strength of the available evidence. We first identify, in the KL-regularized objective, the prefix-level quantity that governs prefix reweighting. We then show that substituting a finite-sample estimate of this quantity is unsafe, and we derive the rule that reweights prefixes only as far as simultaneous confidence bands support, together with the loss that fits the resulting target to the language model.

3.1 KL-REGULARIZED OBJECTIVE AND PREFIX VALUES

For a prompt x and a fixed cutoff, write a response as a prefix s followed by a continuation c, so that $\pi(s, c \mid x) = \pi(s \mid x)\,\pi(c \mid x, s)$, where $\pi(s \mid x)$ is the marginal probability of the prefix and $\pi(c \mid x, s)$ is the conditional distribution over its continuations. Let π_{old} denote the rollout policy, which is held fixed during one rollout-update iteration, and let $R(s, c)$ denote the terminal verifier reward. We consider the KL-regularized objective (Peters et al., 2010; Rafailov et al., 2023; Xiong et al., 2024)

$$\mathcal{J}_x(\pi) = \mathbb{E}_{(s,c)\sim\pi(\cdot,\cdot\mid x)}[R(s,c)] - \frac{1}{\beta} D_{\mathrm{KL}}(\pi(\cdot,\cdot \mid x) \,\|\, \pi_{\mathrm{old}}(\cdot,\cdot \mid x))\,, \qquad \beta > 0, \tag{1}$$

where larger β permits larger departures from the rollout policy. The optimization is over distributions supported by π_{old}.

Optimal continuations and the log-partition function. The chain rule of the KL divergence separates the objective into a continuation term and a prefix term:

$$\mathcal{J}_x(\pi) = \mathbb{E}_{s\sim\pi(\cdot\mid x)}\left[\mathbb{E}_{c\sim\pi(\cdot\mid x,s)}[R(s,c)] - \frac{1}{\beta} D_{\mathrm{KL}}(\pi(\cdot \mid x, s) \,\|\, \pi_{\mathrm{old}}(\cdot \mid x, s))\right] - \frac{1}{\beta} D_{\mathrm{KL}}(\pi(\cdot \mid x) \,\|\, \pi_{\mathrm{old}}(\cdot \mid x))\,.$$

For a fixed prefix marginal $\pi(s \mid x)$, the last term is constant, and each bracketed continuation objective can be maximized separately. Its maximum is expressed through the *partition function* of the prefix under the rollout policy,

$$Z(s \mid x) := \mathbb{E}_{c\sim\pi_{\mathrm{old}}(\cdot\mid x,s)}[e^{\beta R(s,c)}], \tag{2}$$

the expected exponentiated reward, whose logarithm $\log Z(s \mid x)$ is the *log-partition function*. The maximum equals $\beta^{-1} \log Z(s \mid x)$, which we call the *optimal continuation value* of the prefix, and it is attained by the conditional distribution

$$\pi^\star(c \mid x, s) = \frac{\pi_{\mathrm{old}}(c \mid x, s) e^{\beta R(s,c)}}{Z(s \mid x)}.$$

For binary rewards, let

$$\rho(x, s) := \Pr_{c\sim\pi_{\mathrm{old}}(\cdot\mid x,s)}[R(s,c) = 1]$$

denote the success probability of the prefix under the rollout policy, which equals its expected return under π_{old}. Then $Z(s \mid x) = 1 + (e^\beta - 1)\rho(x, s)$.

Marginal objective over prefixes. Pairing a prefix marginal $\pi(\cdot \mid x)$ with the optimal continuation conditional $\pi^\star(c \mid x, s)$ gives the largest objective attainable with that marginal, which defines the marginal objective

$$\begin{aligned}\bar{\mathcal{J}}_x(\pi(\cdot \mid x)) &:= \mathcal{J}_x[\pi(s \mid x)\pi^\star(c \mid x, s)] \\ &= \frac{1}{\beta}\mathbb{E}_{s\sim\pi(\cdot|x)}[\log Z(s \mid x)] - \frac{1}{\beta}D_{\mathrm{KL}}(\pi(\cdot \mid x) \,\|\, \pi_{\mathrm{old}}(\cdot \mid x))\,.\end{aligned} \tag{3}$$

Its maximizer satisfies $\pi^\star(s \mid x) \propto \pi_{\mathrm{old}}(s \mid x)Z(s \mid x)$ (Appendix A). Thus, the log-partition function governs how prefixes are reweighted and must be estimated from continuation samples.

3.2 Reweighting a sampled prefix pool

For each prompt, we generate K independent rollouts under π_{old}, choose a token position t that all K rollouts reach, and truncate each rollout there to obtain the prefixes $s_1, \ldots, s_K$ (Appendix J.2). We write $\mathcal{S} := (s_1, \ldots, s_K)$ for this prefix pool.

Since the prefixes are sampled from $\pi_{\mathrm{old}}(\cdot \mid x)$, the marginal objective of Eq. (3) can be estimated on the pool by importance weighting with the weights $w_i := \pi(s_i \mid x)/\pi_{\mathrm{old}}(s_i \mid x)$, which reparameterize the prefix marginal by the vector $w = (w_1, \ldots, w_K)$. Because the ratio has unit mean under π_{old}, the weights are normalized on the pool, $\sum_i w_i = K$, with feasible set $\mathcal{W} := \{w \in \mathbb{R}_+^K : \sum_i w_i = K\}$. We define $p_i(w) := w_i/K$ as the empirical mass to be assigned to the sampled prefix occurrence s_i via optimization. The unchanged prefix weights $w = \mathbf{1}$ induce $p^0 := p(\mathbf{1})$, with $p_i^0 = 1/K$ for every sampled prefix occurrence. Writing $Z_i := Z(s_i \mid x)$ and $Z := (Z_1, \ldots, Z_K)$, the corresponding empirical marginal objective is

$$\bar{\mathcal{J}}_x^{(K)}(w; Z) := \frac{1}{\beta K}\sum_{i=1}^{K}[w_i \log Z_i - w_i \log w_i]\,, \qquad 0 \log 0 := 0. \tag{4}$$

The gain of the weights w over the unchanged, i.e., uniform baseline is

$$\Delta\bar{\mathcal{J}}(w; Z) := \bar{\mathcal{J}}_x^{(K)}(w; Z) - \bar{\mathcal{J}}_x^{(K)}(\mathbf{1}; Z) = \frac{1}{\beta K}\sum_{i=1}^{K}[(w_i - 1)\log Z_i - w_i \log w_i]\,. \tag{5}$$

The first term rewards moving mass toward prefixes with a larger optimal continuation value $\log Z_i/\beta$, and the move improves the objective only if this gain exceeds the KL cost in the second term.

Plug-in reweighting. If the partition functions were known, the optimal weights would be $w_i^\star = KZ_i/\sum_j Z_j$. Writing $p^\star = p(w^\star)$ for this oracle distribution, the empirical marginal objective of Eq. (4) equals $\beta^{-1}\log\big(\frac{1}{K}\sum_i Z_i\big) - \beta^{-1}D_{\mathrm{KL}}(p(w) \,\|\, p^\star)$, so every weight vector is scored by the KL divergence of $p(w)$ from $p^\star$. In practice, each Z_i is estimated from M independent continuations $c_{i,m} \sim \pi_{\mathrm{old}}(\cdot \mid x, s_i)$ sampled from the prefix,

$$\widehat{Z}_i = \frac{1}{M}\sum_{m=1}^{M} e^{\beta R(s_i, c_{i,m})}, \tag{6}$$

and $\log \widehat{Z}_i$ estimates the log-partition function. The plug-in rule $\widehat{w}_i = K\widehat{Z}_i/\sum_j \widehat{Z}_j$ treats these estimates as exact. Writing $\widehat{p} = p(\widehat{w})$, its true gain is the difference of two divergences from the oracle distribution (Appendix B),

$$\Delta\bar{\mathcal{J}}(\widehat{w}; Z) = \underbrace{\frac{1}{\beta}D_{\mathrm{KL}}(p^0 \| p^\star)}_{\text{available gain}} - \underbrace{\frac{1}{\beta}D_{\mathrm{KL}}(\widehat{p} \| p^\star)}_{\text{estimation error}}\,. \tag{7}$$

The first term measures the gain available from moving the uniform baseline toward the oracle distribution, while the second captures finite-sample estimation error. Plug-in reweighting is harmful when this error exceeds the available gain, a risk that is most relevant when the $\log Z_i$ are similar and the oracle distribution lies close to the baseline. Little improvement is then available, whereas the

plug-in distribution is displaced from the oracle distribution by the sampling noise of the estimates, whose size is set by M rather than by the spread of the values. In the limiting case $Z_1 = \cdots = Z_K$, any nonuniform target incurs a KL cost without gain. Intuitively, equal values mean that no prefix is a better starting point than any other, as when all prefixes succeed with the same probability or, in the extreme, when every continuation succeeds or every one fails, so that the samples carry no information about which prefix to favor. The question is therefore which estimated differences justify reweighting.

3.3 Reweighting under simultaneous confidence bands

TraceBand evaluates each candidate weight vector w over all log-partition values, i.e., optimal continuation values $\beta^{-1} \log Z_i$, consistent with simultaneous confidence bands. We construct intervals $[L_i, U_i]$ for $\log Z_i$ such that, for $0 < \delta < 1$,

$$\mathcal{C} := [L_1, U_1] \times \cdots \times [L_K, U_K], \qquad \Pr(\log Z \in \mathcal{C} \,|\, \mathcal{S}, \pi_{\text{old}}) \geq 1 - \delta. \tag{8}$$

For binary rewards, a two-sided Clopper–Pearson interval $[\rho_i^{\mathrm{L}}, \rho_i^{\mathrm{U}}]$ for the success probability $\rho(x, s_i)$, with failure probability δ/K per prefix (Clopper & Pearson, 1934), is mapped through the relation $Z_i = 1 + (e^\beta - 1)\rho(x, s_i)$:

$$L_i = \log\bigl(1 + (e^\beta - 1)\rho_i^{\mathrm{L}}\bigr), \qquad U_i = \log\bigl(1 + (e^\beta - 1)\rho_i^{\mathrm{U}}\bigr). \tag{9}$$

Monotonicity and a union bound then yield Eq. (8) (Appendix C.1).

Worst-case improvement over the baseline. For a candidate weight vector w, TraceBand evaluates the gain of $p(w)$ over the uniform baseline p^0 under the least favorable log-partition function in $\mathcal{C}$, and selects the weights with the largest worst-case gain:

$$B(w) := \min_{\widetilde{Z}:\, \log \widetilde{Z} \in \mathcal{C}} \Delta\bar{\mathcal{J}}(w; \widetilde{Z}), \qquad w^{\mathrm{TB}} \in \arg\max_{w \in \mathcal{W}} B(w). \tag{10}$$

With the midpoint $m_i = (L_i + U_i)/2$ and the half-width $r_i = (U_i - L_i)/2$ of each interval, the inner minimization gives

$$B(w) = \frac{1}{\beta K} \sum_{i=1}^{K} \left[(w_i - 1) m_i - r_i |w_i - 1| - w_i \log w_i \right]. \tag{11}$$

Thus, uncertainty penalizes departures from the baseline in proportion to $(\beta K)^{-1} \sum_i r_i |w_i - 1|$, with the penalty increasing with both the interval width and the magnitude of the reweighting.

Threshold rule. Maximizing $B(w)$ over $\mathcal{W}$ yields a threshold rule with a common scalar τ, the Lagrange multiplier of the normalization constraint $\sum_i w_i^{\mathrm{TB}} = K$.

Proposition 3.1 (TraceBand threshold rule)**.** *For finite intervals $L_i \leq U_i$, the unique maximizing weights are*

$$w_i^{\mathrm{TB}} = \begin{cases} e^{L_i - \tau}, & L_i > \tau, \\ 1, & L_i \leq \tau \leq U_i, \\ e^{U_i - \tau}, & U_i < \tau, \end{cases} \qquad \sum_{i=1}^{K} w_i^{\mathrm{TB}} = K. \tag{12}$$

The sign of $w_i - 1$ selects the worst-case endpoint, which is L_i when mass is added and U_i when mass is removed. A prefix therefore gains mass only if its entire interval lies above τ, loses mass only if its entire interval lies below τ, and keeps its baseline mass if the interval contains τ. All weights remain unchanged exactly when the intervals share a common point,

$$w^{\mathrm{TB}} = \mathbf{1} \quad \Longleftrightarrow \quad \max_i L_i \leq \min_i U_i, \tag{13}$$

that is, when the bands allow all $\log Z_i$ to be equal. When every interval collapses to $\log Z_i$, the rule recovers the oracle weights $w^\star$. The threshold is found by bisection on $[\min_i L_i, \max_i U_i]$, since the sum of the weights is continuous and non-increasing in τ (Appendix C.2).

Because w^{TB} maximizes $B(w)$ over $\mathcal{W}$, it attains the largest worst-case gain among all feasible weights. On the simultaneous-coverage event,

$$\Delta\bar{\mathcal{J}}(w^{\mathrm{TB}}; Z) \geq B(w^{\mathrm{TB}}) \geq B(\mathbf{1}) = 0. \tag{14}$$

Thus, whenever the confidence bands cover the true log-partition values, the selected reweighting has nonnegative gain relative to the unchanged weights on the fixed sampled-prefix pool.

3.4 FITTING THE REWEIGHTED TARGET

The prefix weights are converted into an update of the language model in two steps. First, a target distribution is defined over the KM sampled responses $(s_i, c_{i,m})$, indexed by the prefix i and the continuation m. Second, normalized model-to-rollout likelihood ratios are fitted to this target by cross-entropy minimization.

The empirical rollout baseline assigns mass $1/(KM)$ to each sampled response. Holding the selected prefix masses w_i^{TB}/K fixed, we choose the joint target P by the same reward–KL tradeoff on this finite sample space:

$$\begin{aligned} \max_{P \geq 0} \quad & \sum_{i,m} P_{i,m} R(s_i, c_{i,m}) - \frac{1}{\beta} \sum_{i,m} P_{i,m} \log(KM P_{i,m}) \\ \text{subject to} \quad & \sum_{m} P_{i,m} = w_i^{\mathrm{TB}}/K, \qquad i = 1, \ldots, K. \end{aligned}$$

With the prefix masses fixed, the chain rule separates the optimization across prefixes. The resulting conditional weights $a_{i,m}$ and joint masses are

$$a_{i,m} = \frac{e^{\beta R(s_i, c_{i,m})}}{\sum_{n=1}^{M} e^{\beta R(s_i, c_{i,n})}}, \qquad P_{i,m} = \frac{w_i^{\mathrm{TB}}}{K} a_{i,m}. \tag{15}$$

Under plug-in prefix weights, the per-prefix normalizers cancel and this target reduces to exponential reward weighting over all sampled responses, a Boltzmann target (Shu et al., 2026).

To connect the target to the policy, recall that $\pi^\star(c \mid x, s_i)/\pi_{\mathrm{old}}(c \mid x, s_i) \propto e^{\beta R(s_i, c)}$. Normalizing these optimal likelihood ratios over the sampled continuations gives exactly $a_{i,m}$, and we likewise seek prefix likelihood ratios proportional to w_i^{TB}. Sequence likelihoods are computed from the next-token probabilities of the model, with model-to-rollout ratios normalized across sampled prefixes for the prefix term and within each prefix for the continuation term:

$$\begin{aligned} q_i^{\mathrm{pre}}(\theta) &= \mathrm{softmax}_i\big(\log \pi_\theta(s_i \mid x) - \log \pi_{\mathrm{old}}(s_i \mid x)\big), \\ q_{i,m}^{\mathrm{cont}}(\theta) &= \mathrm{softmax}_m\big(\log \pi_\theta(c_{i,m} \mid x, s_i) - \log \pi_{\mathrm{old}}(c_{i,m} \mid x, s_i)\big). \end{aligned} \tag{16}$$

The TraceBand loss is the cross entropy between the target $P_{i,m}$ and the fitted joint distribution $q_i^{\mathrm{pre}} q_{i,m}^{\mathrm{cont}}$,

$$\mathcal{L}_{\mathrm{TraceBand}}(\theta; x) = -\sum_{i=1}^{K} \sum_{m=1}^{M} P_{i,m} \log\big(q_i^{\mathrm{pre}}(\theta)\, q_{i,m}^{\mathrm{cont}}(\theta)\big)\,. \tag{17}$$

The rollout policy and the targets are held fixed during fitting and refreshed at the next rollout-update iteration. Appendix D derives the target and loss, and Algorithm 1 summarizes the training procedure.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Tasks and Datasets. For mathematical reasoning, we train on the MATH training split (Hendrycks et al., 2021), excluding Level 1–2 problems, and evaluate on MATH500 (Lightman et al., 2024), AMC 2023,[1] CollegeMath (Tang et al., 2024), MinervaMath (Lewkowycz et al., 2022), and AIME 2024/2025 (Zhang & Math-AI Team, 2024; 2025). For code generation, we train on TACO (Li et al., 2023) and evaluate on HumanEval+, MBPP+ (Liu et al., 2023), LiveCodeBench (Jain et al., 2025), and BigCodeBench (Zhuo et al., 2025). We use binary rewards based on exact final-answer correctness for mathematics and execution-based acceptance for code.

[1] We use the `math-ai/amc23` benchmark.

Table 1: Pass@1 accuracy on mathematical-reasoning and code-generation benchmarks. Results are mean ± standard deviation across three independent training runs. **Bold** and underline indicate the best and second-best results for each model and benchmark, respectively.

(a) Mathematical reasoning

Method	MATH500	AMC23	CollegeMath	MinervaMath	AIME24	AIME25
Qwen2.5-Math-7B						
GRPO	85.6 ± 0.7	59.2 ± 1.4	25.5 ± 0.5	30.0 ± 0.9	15.6 ± 5.1	11.1 ± 5.1
Dr.GRPO	85.6 ± 0.4	57.5 ± 2.5	25.8 ± 0.4	31.1 ± 0.8	15.6 ± 3.8	11.1 ± 6.9
DAPO	86.1 ± 0.5	61.7 ± 2.9	26.4 ± 0.4	32.1 ± 0.6	18.9 ± 5.1	12.2 ± 1.9
VinePPO	86.3 ± 0.7	60.0 ± 2.5	27.6 ± 0.4	33.8 ± 0.4	18.9 ± 6.9	13.3 ± 3.3
GRPO-MA	86.9 ± 0.7	60.8 ± 2.9	28.3 ± 0.2	31.5 ± 1.1	20.0 ± 5.8	13.3 ± 5.8
PPPO	87.2 ± 0.5	62.5 ± 4.3	28.1 ± 0.6	32.6 ± 0.6	**25.6 ± 1.9**	12.2 ± 1.9
TEMPO	86.7 ± 1.0	**67.5 ± 2.5**	27.4 ± 0.2	33.5 ± 1.3	18.9 ± 6.9	15.6 ± 5.1
TraceBand	**89.7 ± 0.6**	**67.5 ± 2.5**	**30.7 ± 0.3**	**36.8 ± 0.4**	**25.6 ± 1.9**	**18.9 ± 1.9**
Qwen3-8B-Base						
GRPO	85.1 ± 0.5	68.3 ± 1.4	28.0 ± 0.5	27.7 ± 1.1	20.0 ± 3.3	20.0 ± 5.8
Dr.GRPO	84.9 ± 0.7	66.7 ± 3.8	28.4 ± 0.4	27.9 ± 1.3	18.9 ± 1.9	21.1 ± 3.8
DAPO	86.1 ± 0.8	71.7 ± 2.9	30.6 ± 0.5	29.9 ± 1.1	23.3 ± 5.8	25.6 ± 5.1
VinePPO	85.5 ± 0.5	72.5 ± 2.5	28.9 ± 0.4	32.0 ± 1.0	22.2 ± 3.8	23.3 ± 3.3
GRPO-MA	87.0 ± 0.4	71.7 ± 1.4	29.5 ± 0.4	28.6 ± 1.2	24.4 ± 3.8	22.2 ± 3.8
PPPO	87.0 ± 0.7	75.0 ± 2.5	29.9 ± 0.6	30.3 ± 1.4	25.6 ± 1.9	**31.1 ± 1.9**
TEMPO	86.3 ± 0.8	74.2 ± 2.9	31.2 ± 0.6	31.4 ± 0.8	24.4 ± 3.8	23.3 ± 3.3
TraceBand	**89.4 ± 0.4**	**77.5 ± 2.5**	**33.9 ± 0.2**	**34.7 ± 0.6**	**28.9 ± 1.9**	28.9 ± 1.9
Llama-3.1-8B-Instruct						
GRPO	63.1 ± 0.6	35.0 ± 2.5	20.9 ± 0.4	22.1 ± 0.4	8.9 ± 6.9	6.7 ± 5.8
Dr.GRPO	62.6 ± 0.7	36.7 ± 5.2	20.7 ± 0.6	23.2 ± 1.1	8.9 ± 6.9	6.7 ± 5.8
DAPO	64.3 ± 0.3	39.2 ± 3.8	24.2 ± 0.2	24.6 ± 1.3	10.0 ± 3.3	10.0 ± 3.3
VinePPO	66.3 ± 0.9	38.3 ± 1.4	22.4 ± 0.5	23.3 ± 1.1	13.3 ± 5.8	6.7 ± 5.8
GRPO-MA	65.5 ± 0.7	40.8 ± 3.8	21.8 ± 0.5	26.0 ± 0.6	11.1 ± 1.9	11.1 ± 6.9
PPPO	66.9 ± 0.7	39.2 ± 3.8	23.7 ± 0.4	26.8 ± 0.7	12.2 ± 3.8	10.0 ± 3.3
TEMPO	**69.9 ± 0.7**	41.7 ± 1.4	22.9 ± 0.6	25.0 ± 1.1	13.3 ± 3.3	10.0 ± 3.3
TraceBand	69.8 ± 0.2	**45.8 ± 1.4**	**27.0 ± 0.3**	**29.5 ± 0.6**	**16.7 ± 3.3**	**15.6 ± 1.9**

(b) Code generation

Method	*Qwen2.5-Coder-7B-Base*				*DeepSeek-Coder-6.7B-Base*			
	Human Eval+	MBPP+	LiveCode Bench	BigCode Bench	Human Eval+	MBPP+	LiveCode Bench	BigCode Bench
GRPO	58.1 ± 0.9	67.2 ± 1.2	30.6 ± 0.8	47.9 ± 0.7	43.9 ± 1.8	60.9 ± 0.8	17.7 ± 0.7	42.9 ± 0.8
Dr.GRPO	58.7 ± 1.5	67.0 ± 0.8	31.0 ± 0.8	47.6 ± 0.8	44.5 ± 1.6	63.5 ± 0.9	18.1 ± 0.9	44.2 ± 0.6
DAPO	60.4 ± 1.6	69.5 ± 1.0	32.0 ± 0.9	48.3 ± 0.7	46.7 ± 0.9	62.5 ± 1.2	18.6 ± 0.8	43.8 ± 0.7
VinePPO	59.8 ± 1.2	69.9 ± 1.2	30.8 ± 0.8	48.7 ± 0.7	45.5 ± 1.5	64.4 ± 0.8	19.5 ± 0.8	44.0 ± 0.8
PPPO	61.2 ± 1.5	68.4 ± 0.9	31.9 ± 0.7	48.6 ± 0.8	46.5 ± 1.5	65.1 ± 0.9	18.5 ± 0.9	45.0 ± 0.6
TEMPO	61.0 ± 1.6	69.2 ± 1.0	31.6 ± 0.8	49.3 ± 0.6	48.0 ± 0.9	63.4 ± 1.3	19.2 ± 0.9	**47.0 ± 0.8**
TraceBand	**63.8 ± 0.9**	**72.2 ± 1.2**	**34.2 ± 0.9**	**51.4 ± 0.7**	**50.8 ± 1.5**	**67.0 ± 0.8**	**21.5 ± 0.7**	46.8 ± 0.8

Baselines. We compare against response-level methods GRPO, Dr.GRPO, and DAPO (Shao et al., 2024; Liu et al., 2025b; Yu et al., 2025), and prefix-level methods VinePPO, PPPO, and TEMPO (Kazemnejad et al., 2025; Sun et al., 2026; Tran et al., 2026), with GRPO-MA additionally included for mathematical reasoning (Wang et al., 2026). For matched-pipeline controls with the same sampling budget and optimization setup, CONTINUATION-ONLY fixes $w_i = 1$, PLUG-IN uses the empirical Boltzmann target, fixed SHRINKAGE interpolates toward Plug-in, WORST-CASE INTERPOLATION maximizes worst-case gain along that direction, MATCHED PLUG-IN matches TRACEBAND's activation and prefix-KL magnitude along the Plug-in direction, and RANDOM-GATED PLUG-IN matches its activation rate by random gating. Formal definitions are given in Appendix G.

Training and Evaluation. For mathematical reasoning, we train Qwen2.5-Math-7B, Qwen3-8B-Base, and Llama-3.1-8B-Instruct (Yang et al., 2024; 2025; Grattafiori et al., 2024); for code generation, we use Qwen2.5-Coder-7B-Base and DeepSeek-Coder-6.7B-Base (Hui et al., 2024; Guo et al.,

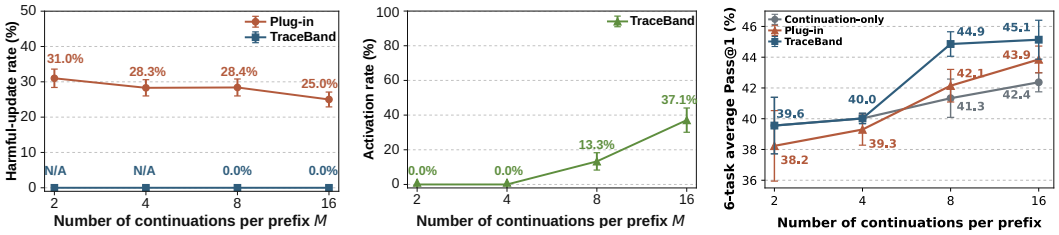

Figure 1: Finite-sample uncertainty across continuation budgets. (a) Harmful-update rate among activated prefix pools using an independent $M_{\text{ref}} = 256$ reference. (b) TRACEBAND activation rate. (c) Six-benchmark average Pass@1. Error bars show standard deviation across three training runs. All experiments use $K = 4$.

Table 2: Exact known-truth validation with $K = 4$, $M = 8$, $\beta = 1$, and $\delta = 0.05$. Harmful-update rates are computed over all sampled prefix pools.

Success Probabilities	Method	Activation (%)	Harmful (%)	Mean True Improvement
$(0.40, 0.45, 0.55, 0.60)$	Plug-in	99.30	84.22	-7.42×10^{-3}
	TRACEBAND	6.98×10^{-2}	1.62×10^{-3}	5.52×10^{-7}
$(0.10, 0.30, 0.70, 0.90)$	Plug-in	100.00	4.26×10^{-2}	3.88×10^{-2}
	TRACEBAND	21.47	$< 10^{-6}$	1.10×10^{-3}

2024). We report Pass@1 on all benchmarks. For TRACEBAND, we use $K = 4$ prefixes at a shared uniformly sampled cutoff and $M = 8$ continuations per prefix, with $\delta = 0.05$ and $\beta = 1$ unless otherwise stated. Verifier budgets are matched across methods; baseline adaptations and compute costs are detailed in Appendices J.3 and I.6. Main comparisons use three independent training seeds with deterministic greedy evaluation. Further implementation details and ablations are provided in Appendices J and I.

5 RESULTS

5.1 OVERALL PERFORMANCE

Table 1 reports Pass@1 accuracy across mathematical reasoning and code generation. TRACEBAND achieves strong performance across multiple model families and both task domains.

5.2 EFFECTS OF FINITE-SAMPLE UNCERTAINTY

Unless otherwise stated, the diagnostics in Sections 5.2–5.3 use Qwen2.5-Math-7B. Using the independent $M_{\text{ref}} = 256$ reference diagnostic (Appendix I.3), we classify an update as harmful when its reference upper bound on improvement is below zero. Figure 1(a)–(b) show that Plug-in can produce harmful updates, whereas TRACEBAND yields uniform prefix weights at $M = 2$ and $M = 4$, reducing exactly to Continuation-only. The exact Bernoulli analysis likewise shows substantially more harmful updates for Plug-in when prefix success probabilities are close, with such updates becoming rare as they separate (Table 2). At $M = 8$, TRACEBAND preserves uniform prefix weights whenever the confidence intervals have a common intersection, preventing unsupported reweighting; additional settings are reported in Appendix H.

Under the default $K = 4$, $M = 8$, and $\delta = 0.05$, TRACEBAND weights require both a $0/8$ and an $8/8$ prefix to be non-uniform; this condition occurs in 13.3% of collected pools. Random-gated Plug-in matches this activation rate, while Matched Plug-in uses the same reweighted pools and matches the magnitude of the prefix reweighting by KL divergence from the uniform baseline. Despite these matches, Matched Plug-in remains constrained to the Plug-in direction, and its prefix-weight vector differs from that of TRACEBAND in 82.4% of reweighted pools (Appendix I.2). Neither control reproduces the TRACEBAND performance gain, showing that activation frequency or reweighting magnitude alone does not explain the gain. At $M = 16$, more success-count patterns

Table 3: Matched-pipeline prefix-update controls on Qwen2.5-Math-7B. Benchmark columns use the default $M = 8$; the last two report six-benchmark averages at $M = 8$ and $M = 16$. Results are mean $\pm$ standard deviation over three runs; bold indicates the best result.

	Pass@1 ($M = 8$)						Avg.	
Method	**MATH500**	**AMC23**	**College Math**	**Minerva Math**	**AIME24**	**AIME25**	**M = 8**	**M = 16**
Continuation-only	87.9 ± 0.8	62.5 ± 2.5	28.3 ± 0.4	33.7 ± 0.2	18.9 ± 5.1	16.7 ± 3.3	41.3 ± 1.2	42.4 ± 0.6
Plug-in	88.1 ± 0.5	63.3 ± 3.8	29.2 ± 0.4	34.4 ± 1.2	21.1 ± 3.8	16.7 ± 3.3	42.1 ± 1.1	43.9 ± 0.9
Shrinkage ($\lambda = 0.5$)	88.5 ± 0.8	65.0 ± 2.5	29.5 ± 0.5	35.0 ± 0.6	22.2 ± 3.8	17.8 ± 5.1	43.0 ± 1.0	43.9 ± 1.0
Worst-case Interpolation	88.9 ± 0.9	65.0 ± 2.5	29.7 ± 0.2	35.7 ± 1.0	21.1 ± 5.1	16.7 ± 3.3	42.8 ± 0.8	44.3 ± 1.1
Matched Plug-in	88.7 ± 0.6	65.0 ± 2.5	29.8 ± 0.5	35.0 ± 0.8	22.2 ± 7.7	16.7 ± 5.8	42.9 ± 0.7	44.2 ± 0.4
Random-gated Plug-in	87.6 ± 0.5	63.3 ± 3.8	28.7 ± 0.4	33.9 ± 0.9	18.9 ± 3.8	15.6 ± 6.9	41.3 ± 1.1	42.8 ± 1.7
TRACEBAND	**89.7 ± 0.6**	**67.5 ± 2.5**	**30.7 ± 0.3**	**36.8 ± 0.4**	**25.6 ± 1.9**	**18.9 ± 1.9**	**44.9 ± 0.8**	**45.1 ± 1.3**

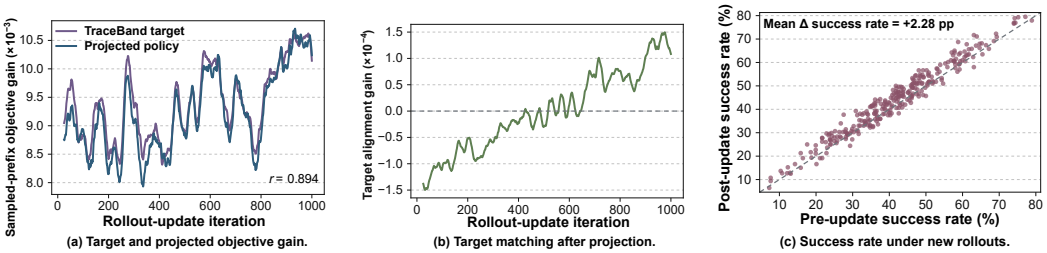

Figure 2: Target-to-policy alignment and fresh-rollout evaluation. (a) Mean target and projected sampled-prefix gains on active pools; curves are smoothed for visualization, while Pearson r is computed before smoothing. (b) $D_{\mathrm{KL}}(p^{\mathrm{TB}} \| p^0) - D_{\mathrm{KL}}(p^{\mathrm{TB}} \| q_\theta)$. (c) Pre/post success rates on 300 fixed MATH-test prompts; the dashed line denotes equality.

yield non-uniform TRACEBAND weights and the larger continuation budget reduces finite-sample uncertainty in Plug-in estimates. Consistent with this, the performance gap narrows while TRACEBAND retains an advantage over the matched controls (Table 3).

5.3 TARGET-TO-POLICY ALIGNMENT

TRACEBAND specifies a target distribution over sampled prefixes, whereas optimization updates a parameterized LLM. We test whether the fitted policy realizes this target and whether the resulting change transfers to fresh continuations. For each prefix pool, let π_{old} denote the rollout policy and π_θ the policy after three optimizer updates on the fixed batch.

On pools with $w^{\mathrm{TB}} \neq \mathbf{1}$, let $q_{\theta,i} \propto \pi_\theta(s_i \mid x)/\pi_{\mathrm{old}}(s_i \mid x)$ and $w_i^{\mathrm{proj}} := K q_{\theta,i}$. Using the same pool-level $\widehat{Z}$ from Eq. (6), the projected gains closely track the TRACEBAND target gains throughout training ($r = 0.894$; Figure 2(a)). With $p_i^{\mathrm{TB}} = w_i^{\mathrm{TB}}/K$ and $p_i^0 = 1/K$, Figure 2(b) further shows that q_θ becomes closer to p^{TB} than to p^0 later in training.

Finally, on 300 fixed prompts from the MATH test split, 512 fresh rollouts from each policy show positive empirical changes on 74.3% of prompts and a mean gain of 2.28 percentage points (95% prompt-level bootstrap CI: 1.95–2.62; Figure 2(c)). These diagnostics show that the fitted policy moves toward the TRACEBAND target and improves fresh-rollout success.

6 CONCLUSION

We study finite-sample prefix reweighting, where uncertain log-partition estimates can induce harmful updates. TRACEBAND reweights only when simultaneous confidence bands support a positive worst-case improvement and otherwise preserves the original prefix weights, so that the reweighting has nonnegative gain under simultaneous coverage. Across mathematical reasoning and code generation, TRACEBAND achieves strong downstream performance, and the fitted policy also yields gains on fresh rollouts.

AI USE STATEMENT

In this work, we used generative AI tools to improve manuscript clarity, assist with literature retrieval, sanity-check mathematical derivations, and review experimental code. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

We provide complete derivations and proofs for the theoretical results in Appendices A–D, and summarize the TRACEBAND training procedure in Algorithm 1. Detailed training hyperparameters, verifier implementations, prefix and continuation sampling procedures, and evaluation settings are reported in Appendix J. Baseline-specific implementations and verifier-budget matching are described in Appendix J.3. Additional diagnostics, ablations, and compute measurements are provided in Appendix I. Anonymous source code and experiment configurations are provided in the supplementary material.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Charles J Clopper and Egon S Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 1934.

Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, 2016.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in large language models via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

Baizhou Huang and Xiaojun Wan. PROS: Towards compute-efficient RLVR via rollout prefix reuse. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lz1SRTcnUb.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Doo Hwan Hwang and Kee-Eung Kim. PS-PPO: Prefix-sampling PPO for critic-free RLHF. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.29758.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron C. Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in RL training of llms. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, 2025.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay 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 *Advances in Neural Information Processing Systems*, volume 35, 2022.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset, 2023.

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.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, 2023.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

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

Kimia Nadjahi, Romain Laroche, and Rémi Tachet des Combes. Safe policy improvement with soft baseline bootstrapping. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2019. URL https://arxiv.org/abs/1907.05079.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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. *arXiv preprint arXiv:2402.03300*, 2024.

Yao Shu, Chenxing Wei, Hongbin Lin, Shuang Qiu, and Hui Xiong. Reference-sampled boltzmann projection for KL-regularized RLVR: Target-matched weighted SFT, finite one-shot gaps, and policy mirror descent, 2026.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems*, 2025.

Yiliu Sun, Zicheng Zhao, Yang Wei, Yanfang Zhang, and Chen Gong. Well begun, half done: Reinforcement learning with prefix optimization for llm reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. MathScale: Scaling instruction tuning for mathematical reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 21–27 Jul 2024.

Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Hieu Tran, Zonghai Yao, and Hong Yu. Exploiting tree structure for credit assignment in reinforcement learning with large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*. Association for Computational Linguistics, 2026.

Hongcheng Wang, Yinuo Huang, Sukai Wang, Guanghui Ren, and Hao Dong. Why tree-style branching matters for thought advantage estimation in GRPO. In *Forty-third International Conference on Machine Learning*, 2026.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*. Association for Computational Linguistics, 2024.

Can Xie, Ruotong Pan, Xiangyu Wu, Yunfei Zhang, Jiayi Fu, Tingting Gao, and Guorui Zhou. Unlocking exploration in RLVR: Uncertainty-aware advantage shaping for deeper reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 19057–19076. Association for Computational Linguistics, 2026.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for RLHF under KL-constraint. In *Forty-first International Conference on Machine Learning*, 2024.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2025.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. ACECODER: Acing coder RL via automated test-case synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2025.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2024, 2024. URL `https://huggingface.co/datasets/math-ai/aime24`.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2025, 2025. URL `https://huggingface.co/datasets/math-ai/aime25`.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen GONG, James Hoang, Armel Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025.

A Derivation of the Exact KL-Regularized Target

We derive the continuation and prefix targets associated with Eq. (1). Fix a prompt x, a cutoff, and the rollout policy π_{old}. The reward is bounded and $\beta > 0$. Candidate joint distributions are supported by π_{old}; all conditional expressions below concern prefixes with positive rollout probability. This is a distribution-level optimization, before fitting a parameterized language model.

KL chain rule. The factorizations

$$\pi(s, c \mid x) = \pi(s \mid x)\pi(c \mid x, s), \qquad \pi_{\text{old}}(s, c \mid x) = \pi_{\text{old}}(s \mid x)\pi_{\text{old}}(c \mid x, s)$$

give

$$\begin{aligned} &D_{\text{KL}}(\pi(\cdot, \cdot \mid x) \,\|\, \pi_{\text{old}}(\cdot, \cdot \mid x)) \\ &\quad = D_{\text{KL}}(\pi(\cdot \mid x) \,\|\, \pi_{\text{old}}(\cdot \mid x)) + \mathbb{E}_{s\sim\pi(\cdot|x)} D_{\text{KL}}(\pi(\cdot \mid x, s) \,\|\, \pi_{\text{old}}(\cdot \mid x, s)) \,. \end{aligned}$$

Substitution into Eq. (1) yields

$$\begin{aligned} \mathcal{J}_x(\pi) = \mathbb{E}_{s\sim\pi(\cdot|x)} \left[\mathbb{E}_{c\sim\pi(\cdot|x,s)}[R(s, c)] - \frac{1}{\beta} D_{\text{KL}}(\pi(\cdot \mid x, s) \,\|\, \pi_{\text{old}}(\cdot \mid x, s)) \right] \\ - \frac{1}{\beta} D_{\text{KL}}(\pi(\cdot \mid x) \,\|\, \pi_{\text{old}}(\cdot \mid x)) \,. \end{aligned}$$

For a fixed prefix distribution, the last term is constant and each continuation conditional can be optimized separately.

Optimal continuation conditional. Fix a prefix s. Recall the conditional partition function from Eq. (2),

$$Z(s \mid x) = \mathbb{E}_{c\sim\pi_{\text{old}}(\cdot|x,s)}[e^{\beta R(s,c)}],$$

and the associated optimal continuation distribution below. Bounded rewards ensure $0 < Z(s \mid x) < \infty$. Define

$$\pi^\star(c \mid x, s) = \frac{\pi_{\text{old}}(c \mid x, s) e^{\beta R(s,c)}}{Z(s \mid x)}. \tag{18}$$

This distribution is normalized. The log-partition function is $\log Z(s \mid x)$; scaling it by β^{-1} gives the optimal KL-regularized continuation value, as the identity below makes explicit. Moreover, $\log(\pi^\star(c \mid x, s)/\pi_{\text{old}}(c \mid x, s)) = \beta R(s, c) - \log Z(s \mid x)$. Expanding the KL divergence to $\pi^\star$ therefore gives

$$\begin{aligned} &\mathbb{E}_{c\sim\pi(\cdot|x,s)}[R(s, c)] - \frac{1}{\beta} D_{\text{KL}}(\pi(\cdot \mid x, s) \,\|\, \pi_{\text{old}}(\cdot \mid x, s)) \\ &\qquad = \frac{1}{\beta} \log Z(s \mid x) - \frac{1}{\beta} D_{\text{KL}}(\pi(\cdot \mid x, s) \,\|\, \pi^\star(\cdot \mid x, s)) \,. \end{aligned} \tag{19}$$

The second term is non-positive and vanishes at $\pi^\star$. Consequently, the maximum continuation objective is $\beta^{-1} \log Z(s \mid x)$, not the rollout-policy expected return. For binary rewards, the latter is

$$\rho(x, s) = \Pr_{c\sim\pi_{\text{old}}(\cdot|x,s)}[R(s, c) = 1], \qquad Z(s \mid x) = 1 + (e^\beta - 1)\rho(x, s).$$

Marginal objective and prefix target. Pairing any prefix distribution with Eq. (18) gives

$$\bar{\mathcal{J}}_x(\pi(\cdot \mid x)) = \frac{1}{\beta} \mathbb{E}_{s\sim\pi(\cdot|x)}[\log Z(s \mid x)] - \frac{1}{\beta} D_{\text{KL}}(\pi(\cdot \mid x) \,\|\, \pi_{\text{old}}(\cdot \mid x)) \,,$$

which is Eq. (3). Define

$$\pi^\star(s \mid x) = \frac{\pi_{\text{old}}(s \mid x) Z(s \mid x)}{\sum_u \pi_{\text{old}}(u \mid x) Z(u \mid x)}. \tag{20}$$

Expanding the prefix KL to this distribution yields

$$\bar{\mathcal{J}}_x(\pi(\cdot \mid x)) = \frac{1}{\beta} \log \left(\sum_u \pi_{\text{old}}(u \mid x) Z(u \mid x) \right) - \frac{1}{\beta} D_{\text{KL}}(\pi(\cdot \mid x) \,\|\, \pi^\star(\cdot \mid x)) \,.$$

Thus Eq. (20) maximizes the marginal objective. Together, the two target factors give

$$\pi^\star(s, c \mid x) = \frac{\pi_{\text{old}}(s, c \mid x) e^{\beta R(s,c)}}{\sum_u \pi_{\text{old}}(u \mid x) Z(u \mid x)}.$$

For an arbitrary candidate joint policy, rather than one paired with optimal continuation conditionals, Eq. (19) instead implies

$$\begin{aligned}\mathcal{J}_x(\pi) = \bar{\mathcal{J}}_x(\pi(\cdot \mid x)) \\ - \frac{1}{\beta} \mathbb{E}_{s \sim \pi(\cdot|x)} D_{\text{KL}}(\pi(\cdot \mid x, s) \,\|\, \pi^\star(\cdot \mid x, s)) \,.\end{aligned}$$

The marginal objective therefore evaluates the prefix distribution after analytically optimizing the continuation conditionals. This analytical elimination is not an inner optimization performed during rollout, and a guarantee for the marginal objective does not by itself guarantee improvement after parameterized joint-policy fitting.

B Derivation of the Sampled-Prefix Plug-In Decomposition

We derive Eq. (5) and Eq. (7) on a fixed pool of K sampled prefix occurrences. The empirical baseline is $p_i^0 = 1/K$; it is not the rollout policy conditioned on the set of distinct observed strings. For $w \in \mathcal{W} = \{w \in \mathbb{R}_+^K : \sum_i w_i = K\}$, let $p_i(w) = w_i/K$ and $Z_i = Z(s_i \mid x)$. The empirical marginal objective is

$$\bar{\mathcal{J}}_x^{(K)}(w; Z) = \frac{1}{\beta K} \sum_{i=1}^{K} \left[w_i \log Z_i - w_i \log w_i \right], \qquad 0 \log 0 := 0.$$

Here the second term is $-\beta^{-1} D_{\text{KL}}(p(w) \| p^0)$. Subtracting its value at $w = \mathbf{1}$ gives

$$\Delta\bar{\mathcal{J}}(w; Z) = \bar{\mathcal{J}}_x^{(K)}(w; Z) - \bar{\mathcal{J}}_x^{(K)}(\mathbf{1}; Z) = \frac{1}{\beta K} \sum_i \left[(w_i - 1) \log Z_i - w_i \log w_i \right],$$

as in Eq. (5).

Oracle distribution and general gap identity. Define $p_i^\star = Z_i / \sum_j Z_j$. Expanding the KL divergence gives

$$D_{\text{KL}}(p(w) \| p^\star) = \frac{1}{K} \sum_i w_i \log w_i - \frac{1}{K} \sum_i w_i \log Z_i + \log \left(\frac{1}{K} \sum_i Z_i \right).$$

Hence

$$\bar{\mathcal{J}}_x^{(K)}(w; Z) = \frac{1}{\beta} \log \left(\frac{1}{K} \sum_i Z_i \right) - \frac{1}{\beta} D_{\text{KL}}(p(w) \| p^\star).$$

This proves that the oracle weights are $w_i^\star = K Z_i / \sum_j Z_j$. Subtracting the identity at the unchanged weights yields

$$\Delta\bar{\mathcal{J}}(w; Z) = \frac{1}{\beta} \left[D_{\text{KL}}(p^0 \| p^\star) - D_{\text{KL}}(p(w) \| p^\star) \right].$$

With $\widehat{w}_i = K \widehat{Z}_i / \sum_j \widehat{Z}_j$ and $\widehat{p} = p(\widehat{w})$, this is precisely Eq. (7).

Interpretation of the plug-in gap. Eq. (7) separates the plug-in update into the oracle improvement available on the realized pool, $\beta^{-1} D_{\text{KL}}(p^0 \| p^\star)$, and the loss caused by estimating the target distribution, $\beta^{-1} D_{\text{KL}}(\widehat{p} \| p^\star)$. Both terms are nonnegative. Hence the plug-in update is harmful exactly when the estimation loss exceeds the available oracle improvement.

If all partition-function estimates differ from the true values by the same multiplicative factor, $\widehat{Z}_i = a Z_i$ for every i and some $a > 0$, then normalization cancels that factor and $\widehat{p} = p^\star$, so the estimation

loss is zero. If instead all true values are equal, $Z_1 = \cdots = Z_K$, then $p^\star = p^0$ and every feasible reweighting satisfies

$$\Delta\bar{\mathcal{J}}(w; Z) = -\frac{1}{\beta} D_{\mathrm{KL}}(p(w) \| p^0) \le 0,$$

with strict inequality whenever $p(w) \neq p^0$. This proves the equal-value statement in the main text without introducing additional gap notation used only in this appendix.

C TraceBand Derivations

C.1 Confidence Intervals for Log-Partition Values

Let $\mathcal{S}$ denote the realized indexed prefix pool and common cutoff. Condition on $\mathcal{S}$ and the frozen rollout policy. For each sampled prefix occurrence, TraceBand requires an interval for $\log Z_i = \log Z(s_i \mid x)$. The M fresh continuation draws estimate Z_i as in Eq. (6). We construct $[L_i, U_i]$ for $\log Z_i$ with failure probability at most δ/K. A union bound gives

$$\Pr(\exists i : \log Z_i \notin [L_i, U_i] \,|\, \mathcal{S}, \pi_{\mathrm{old}}) \le \sum_i \Pr(\log Z_i \notin [L_i, U_i] \,|\, \mathcal{S}, \pi_{\mathrm{old}}) \le \delta.$$

Consequently, with the log-partition uncertainty set

$$\mathcal{C} := \prod_{i=1}^{K} [L_i, U_i],$$

we have

$$\Pr(\log Z \in \mathcal{C} \,|\, \mathcal{S}, \pi_{\mathrm{old}}) \ge 1 - \delta,$$

which is Eq. (8). The union bound does not require distinct prefix strings or independence between different prefixes' intervals. It uses valid conditional coverage for each occurrence; within each occurrence, the continuation samples must follow the stated i.i.d. sampling law.

Binary rewards. Write

$$\rho_i := \Pr_{c \sim \pi_{\mathrm{old}}(\cdot \mid x, s_i)} [R(s_i, c) = 1], \qquad S_i := \sum_{m=1}^{M} R(s_i, c_{i,m}).$$

For fixed M, $S_i \mid \mathcal{S}, \pi_{\mathrm{old}} \sim \mathrm{Binomial}(M, \rho_i)$. We use a two-sided Clopper–Pearson interval $[\rho_i^{\mathrm{L}}, \rho_i^{\mathrm{U}}]$ with total failure probability δ/K, assigning $\delta/(2K)$ to each tail (Clopper & Pearson, 1934). The lower endpoint is 0 when $S_i = 0$; otherwise it solves

$$\sum_{k=S_i}^{M} \binom{M}{k} (\rho_i^{\mathrm{L}})^k (1 - \rho_i^{\mathrm{L}})^{M-k} = \frac{\delta}{2K}.$$

The upper endpoint is 1 when $S_i = M$; otherwise it solves

$$\sum_{k=0}^{S_i} \binom{M}{k} (\rho_i^{\mathrm{U}})^k (1 - \rho_i^{\mathrm{U}})^{M-k} = \frac{\delta}{2K}.$$

Because $Z_i = 1 + (e^\beta - 1)\rho_i$ and both this map and $\log$ are increasing, the transformed interval is

$$L_i = \log\big(1 + (e^\beta - 1)\rho_i^{\mathrm{L}}\big), \qquad U_i = \log\big(1 + (e^\beta - 1)\rho_i^{\mathrm{U}}\big),$$

as in Eq. (9). This is the fixed-sample construction used here, not an anytime-valid confidence sequence for data-dependent stopping.

C.2 Robust Improvement and the Threshold Rule

We derive Eq. (11) and prove Proposition 3.1. For a fixed w, set $v_i = \log \widetilde{Z}_i$. The product uncertainty set in Eq. (10) gives

$$B(w) = \frac{1}{\beta K} \sum_i \left[\min_{v_i \in [L_i, U_i]} (w_i - 1) v_i - w_i \log w_i \right].$$

The coordinate-wise minimum is

$$\min_{v_i \in [L_i, U_i]} (w_i - 1) v_i = \begin{cases} (w_i - 1) L_i, & w_i > 1, \\ 0, & w_i = 1, \\ (w_i - 1) U_i, & 0 \le w_i < 1. \end{cases}$$

Equivalently, with $m_i = (L_i + U_i)/2$ and $r_i = (U_i - L_i)/2$, it is $(w_i - 1)m_i - r_i |w_i - 1|$. This proves Eq. (11). The uncertainty penalty acts on deviation from the baseline, not on the absolute prefix score.

Proof of Proposition 3.1. Ignoring the positive factor $(\beta K)^{-1}$, maximize $\sum_i \phi_i(w_i)$ subject to $w_i \ge 0$ and $\sum_i w_i = K$, where

$$\phi_i(w_i) = \begin{cases} (w_i - 1) U_i - w_i \log w_i, & 0 \le w_i \le 1, \\ (w_i - 1) L_i - w_i \log w_i, & w_i \ge 1. \end{cases}$$

Each function is strictly concave: away from one, $\phi_i''(w_i) = -1/w_i < 0$, and the slope decreases from $U_i - 1$ to $L_i - 1$ at one. The objective is continuous on the compact feasible simplex and has a unique optimizer. Since the right derivative at zero is $+\infty$ for finite endpoints, every optimal coordinate is positive.

Use the Lagrangian $\sum_i \phi_i(w_i) + \lambda(\sum_i w_i - K)$ and set $\tau = 1 - \lambda$. For $w_i > 1$, stationarity gives $0 = L_i - \log w_i - 1 + \lambda$, hence $w_i = e^{L_i - \tau}$, which is greater than one exactly when $L_i > \tau$. For $0 < w_i < 1$, it gives $w_i = e^{U_i - \tau}$, which is less than one exactly when $U_i < \tau$. At $w_i = 1$, the left and right Lagrangian derivatives are $U_i - \tau$ and $L_i - \tau$. The optimality condition is therefore $L_i \le \tau \le U_i$. Combining the cases gives Eq. (12).

Let $F(\tau) = \sum_i w_i(\tau)$. It is continuous and non-increasing, with $F(\min_i L_i) \ge K$ and $F(\max_i U_i) \le K$. Thus a normalizing threshold can be found by bisection on $[\min_i L_i, \max_i U_i]$. A flat set of normalizing thresholds can occur only when all weights equal one; the maximizing weight vector remains unique. □

No-change characterization. If the intervals share a common value, any threshold in their intersection gives $w_i = 1$ for every i. Conversely, $w^{\mathrm{TB}} = \mathbf{1}$ requires a threshold belonging to every interval. Thus

$$w^{\mathrm{TB}} = \mathbf{1} \quad \Longleftrightarrow \quad \bigcap_i [L_i, U_i] \neq \varnothing \quad \Longleftrightarrow \quad \max_i L_i \le \min_i U_i,$$

proving Eq. (13). This is a consequence of the robust optimization, not a separate heuristic gate.

Recovery of the oracle weights. When $L_i = U_i = \log Z_i$, every branch becomes $w_i = Z_i e^{-\tau}$. Normalization gives $e^{-\tau} = K / \sum_j Z_j$, so $w_i^{\mathrm{TB}} = K Z_i / \sum_j Z_j = w_i^\star$.

C.3 Finite-Sample Improvement and Oracle-Gap Bound

When the confidence intervals simultaneously cover the log-partition values at all sampled prefixes, $\log Z \in \mathcal{C}$. Therefore, for every feasible w,

$$\Delta\bar{\mathcal{J}}(w; Z) \ge \min_{\widetilde{Z}:\, \log \widetilde{Z} \in \mathcal{C}} \Delta\bar{\mathcal{J}}(w; \widetilde{Z}) = B(w).$$

The unchanged weights are feasible and satisfy $B(\mathbf{1}) = 0$, because $\Delta\bar{\mathcal{J}}(\mathbf{1}; \widetilde{Z}) = 0$ for every $\widetilde{Z}$. Optimality of w^{TB} therefore gives

$$\Delta\bar{\mathcal{J}}(w^{\mathrm{TB}}; Z) \ge B(w^{\mathrm{TB}}) \ge B(\mathbf{1}) = 0,$$

which yields Eq. (14).

For completeness, we also bound the gap to the oracle weights on the same sampled-prefix pool. Since $w^\star$ maximizes $\Delta\bar{\mathcal{J}}(\cdot; Z)$ over the same feasible set,

$$0 \le \Delta\bar{\mathcal{J}}(w^\star; Z) - \Delta\bar{\mathcal{J}}(w^{\mathrm{TB}}; Z).$$

Moreover, $\Delta\bar{\mathcal{J}}(w^{\mathrm{TB}}; Z) \ge B(w^{\mathrm{TB}}) \ge B(w^\star)$, and hence

$$\begin{aligned}
0 &\le \Delta\bar{\mathcal{J}}(w^\star; Z) - \Delta\bar{\mathcal{J}}(w^{\mathrm{TB}}; Z) \\
&\le \Delta\bar{\mathcal{J}}(w^\star; Z) - B(w^\star) \\
&= \frac{1}{\beta K}\left[\sum_{i: w_i^\star > 1} (w_i^\star - 1)(\log Z_i - L_i) + \sum_{i: w_i^\star < 1} (1 - w_i^\star)(U_i - \log Z_i)\right] \\
&\le \frac{1}{\beta K}\sum_i |w_i^\star - 1|(U_i - L_i) \\
&\le \frac{2}{\beta}\max_i (U_i - L_i). \qquad (21)
\end{aligned}$$

The penultimate inequality uses $L_i \le \log Z_i \le U_i$, and the last follows from $\sum_i |w_i^\star - 1| \le \sum_i (w_i^\star + 1) = 2K$.

The coverage event provides the lower bound simultaneously for every feasible w, so the same continuation samples may be used to construct the intervals and select w^{TB}. Both the improvement bound and the oracle-gap bound hold with conditional probability at least $1 - \delta$ over the fresh continuations, conditional on the realized sampled-prefix pool. These results apply to the marginal objective on the fixed sampled-prefix pool.

D Empirical Joint Target and Normalized-Ratio Fitting

Continuation and joint targets. Fix the TraceBand prefix masses $p_i^{\mathrm{TB}} := w_i^{\mathrm{TB}}/K$. On the sampled index space (i, m), the empirical rollout baseline is uniform with mass $1/(KM)$. The finite-sample target in the main text solves

$$\max_{P \ge 0} \sum_{i,m} P_{i,m} R(s_i, c_{i,m}) - \frac{1}{\beta}\sum_{i,m} P_{i,m}\log(KMP_{i,m}) \quad \text{subject to} \quad \sum_m P_{i,m} = p_i^{\mathrm{TB}}.$$

Write $P_{i,m} = p_i^{\mathrm{TB}} a_{i,m}$, where $\sum_m a_{i,m} = 1$. Substitution separates all terms that depend on $a_i = (a_{i,1}, \ldots, a_{i,M})$: for each prefix i, maximizing over a_i is equivalent to

$$\max_{a_i \in \Delta_M} \sum_m a_{i,m} R(s_i, c_{i,m}) - \frac{1}{\beta} D_{\mathrm{KL}}(a_i \,\|\, u_M), \qquad u_M(m) = \frac{1}{M}.$$

This is the empirical analogue of Eq. (18), so the optimizer is

$$a_{i,m} = \frac{M^{-1} e^{\beta R(s_i, c_{i,m})}}{\sum_n M^{-1} e^{\beta R(s_i, c_{i,n})}} = \frac{e^{\beta R(s_i, c_{i,m})}}{\sum_n e^{\beta R(s_i, c_{i,n})}}.$$

Therefore

$$P_{i,m} = p_i^{\mathrm{TB}} a_{i,m} = \frac{w_i^{\mathrm{TB}}}{K} a_{i,m}, \qquad \sum_m P_{i,m} = p_i^{\mathrm{TB}}, \qquad \sum_{i,m} P_{i,m} = 1,$$

which is Eq. (15).

Equivalently, write $b_{i,m} = 1/(KM)$, $p_i^0 = 1/K$, and $u_M(m) = 1/M$. With $P_{i,m} = p_i^{\mathrm{TB}} v_{i,m}$, the KL chain rule on the sampled index space gives

$$D_{\mathrm{KL}}(P \| b) = D_{\mathrm{KL}}(p^{\mathrm{TB}} \| p^0) + \sum_i p_i^{\mathrm{TB}} D_{\mathrm{KL}}(v_i \| u_M).$$

Thus, once the prefix masses are fixed, each conditional separately maximizes $\sum_m v_{i,m} R_{i,m} - \beta^{-1} D_{\mathrm{KL}}(v_i \| u_M)$, whose maximizer is a_i. In particular, using $\log a_{i,m} = \beta R_{i,m} - \log(M \widehat{Z}_i)$, the finite-sample objective above can be rewritten for any feasible $P_{i,m} = p_i^{\mathrm{TB}} v_{i,m}$ as

$$\frac{1}{\beta} \sum_i p_i^{\mathrm{TB}} \log \widehat{Z}_i - \frac{1}{\beta} D_{\mathrm{KL}}(p^{\mathrm{TB}} \| p^0) - \frac{1}{\beta} \sum_i p_i^{\mathrm{TB}} D_{\mathrm{KL}}(v_i \| a_i).$$

Therefore the maximum is attained at $P_{i,m} = p_i^{\mathrm{TB}} a_{i,m}$. A row with $p_i^{\mathrm{TB}} = 0$ contributes nothing and has $P_{i,m} = 0$.

These are probabilities on sampled indices, not the original policy's probabilities of the observed strings; repeated strings contribute through their separate sampled occurrences. Because continuations are drawn from π_{old}, sums weighted by a_i are self-normalized importance-sampling estimates of expectations under $\pi^\star(\cdot \mid x, s_i)$. Consequently, the empirical optimum does not assert $a_{i,m} = \pi^\star(c_{i,m} \mid x, s_i)$, nor does it extend the marginal-objective improvement bound to the fitted joint policy.

Under plug-in prefix weights, the common continuation count M gives

$$\frac{\widehat{w}_i}{K} = \frac{\widehat{Z}_i}{\sum_j \widehat{Z}_j} = \frac{\sum_n e^{\beta R(s_i, c_{i,n})}}{\sum_{j,n} e^{\beta R(s_j, c_{j,n})}}.$$

The within-prefix normalizer then cancels:

$$P_{i,m}^{\mathrm{plug}} = \frac{e^{\beta R(s_i, c_{i,m})}}{\sum_{j,n} e^{\beta R(s_j, c_{j,n})}}.$$

This establishes the global Boltzmann-weighting identity stated in the main text. It is an identity of sampled targets, not an equivalence of all fitting algorithms.

Normalized likelihood ratios. The prefix and continuation targets have the same mathematical form. Consider generic sampled occurrences $y_1, \ldots, y_N$ with positive rollout probabilities and a normalized target $t = (t_1, \ldots, t_N)$ on their indices. Define

$$q_j(\theta) = \frac{\pi_\theta(y_j)/\pi_{\mathrm{old}}(y_j)}{\sum_r \pi_\theta(y_r)/\pi_{\mathrm{old}}(y_r)}.$$

For a normalized target t,

$$-\sum_j t_j \log q_j(\theta) = -\sum_j t_j \log t_j + D_{\mathrm{KL}}(t \| q_\theta).$$

Applied separately to prefixes and continuations, this gives exactly q_i^{pre} and $q_{i,m}^{\mathrm{cont}}$ in Eq. (16). The ideal continuation policy exactly matches this normalized-ratio target on the observed samples: substituting $\pi_\theta(\cdot \mid x, s_i) = \pi^\star(\cdot \mid x, s_i)$ gives

$$q_{i,m}^{\mathrm{cont}} = \frac{e^{\beta R_{i,m}}/Z_i}{\sum_n e^{\beta R_{i,n}}/Z_i} = a_{i,m}.$$

Accordingly, fitting a_i matches relative likelihood changes from the rollout policy rather than setting raw continuation probabilities equal to a_i.

The product

$$Q_{i,m}(\theta) := q_i^{\mathrm{pre}}(\theta) q_{i,m}^{\mathrm{cont}}(\theta)$$

is a normalized distribution over sampled response occurrences. Its cross entropy from the joint target is

$$\begin{aligned} -\sum_{i,m} P_{i,m} \log Q_{i,m}(\theta) &= -\sum_i \frac{w_i^{\mathrm{TB}}}{K} \left[\log q_i^{\mathrm{pre}}(\theta) + \sum_m a_{i,m} \log q_{i,m}^{\mathrm{cont}}(\theta) \right] \\ &= \mathcal{L}_{\mathrm{TraceBand}}(\theta; x), \end{aligned}$$

proving Eq. (17). Equivalently, up to the fixed target entropy, the loss is

$$D_{\mathrm{KL}}(P\|Q_\theta) = D_{\mathrm{KL}}(p^{\mathrm{TB}}\|q_\theta^{\mathrm{pre}}) + \sum_i p_i^{\mathrm{TB}} D_{\mathrm{KL}}(a_i\|q_{i,\theta}^{\mathrm{cont}}).$$

The same prefix mass therefore weights both components of one joint target. If the parameterized family can realize both factors jointly, exact matching attains the target entropy; otherwise the objective defines a forward-KL approximation on the sampled occurrences. Optimization need not attain that minimum. The rollout policy, rewards, confidence bands, and target probabilities are held fixed during fitting.

Scope of the fitting surrogate. For a finite sampled pool, exact ratio-target matching $q_j(\theta) = t_j$ implies only

$$\frac{\pi_\theta(y_j)}{\pi_{\mathrm{old}}(y_j)} = \kappa t_j, \qquad j = 1, \ldots, N,$$

for a common $\kappa > 0$. Normalization removes this common scale and does not fix the total probability assigned to the sampled set or behavior outside it. Likewise, an unchanged prefix target does not freeze actual prefix probabilities under shared-parameter continuation updates. The improvement bound for the empirical target therefore does not automatically transfer to the fitted policy, as stated in the main text.

If two occurrences have the same prefix string, their prefix likelihood ratios are equal under every θ. Occurrence-specific targets can nevertheless differ because they use different continuation samples; exact fitting is then constrained by this shared representation. The target-level confidence and improvement statements remain valid on the occurrence pool and do not assert realizability of every such target.

E TraceBand Algorithm

Algorithm 1 TraceBand update procedure.

Require: Current policy π_θ, prompt batch $\mathcal{B}$, K, M, β, δ

Set $\pi_{\mathrm{old}} \leftarrow \pi_\theta$ and keep π_{old} frozen throughout this rollout-update iteration
for each prompt $x \in \mathcal{B}$ **do**
  Generate K independent initial rollouts under π_{old}
  Choose a token position t available in every initial rollout and truncate all rollouts at t, obtaining $s_1, \ldots, s_K$ (Appendix J.2)
  for $i = 1, \ldots, K$ **do**
    Sample M fresh, independent continuations $c_{i,m} \sim \pi_{\mathrm{old}}(\cdot \mid x, s_i)$ and evaluate $R(s_i, c_{i,m})$
  end for
  Construct simultaneous intervals $[L_i, U_i]_{i=1}^K$ for $\log Z_i$ at level $1-\delta$ using Eq. (9)
  Compute $(w_i^{\mathrm{TB}})_{i=1}^K$ using Eq. (12), with $\sum_i w_i^{\mathrm{TB}} = K$
  for $i = 1, \ldots, K$ **do**
    Compute continuation probabilities $a_{i,m}$ and joint target $P_{i,m}$ using Eq. (15)
  end for
end for
Update θ for three optimizer updates by minimizing $\mathcal{L}_{\mathrm{TraceBand}}$ in Eq. (17), while keeping the sampled pools and target probabilities fixed.

F Activation under Finite Binary Samples

Consider the occurrence-level pool with continuation budget M and failure probability δ/K for each occurrence. Under the simultaneous two-sided Clopper–Pearson construction, the confidence interval for zero successes has upper endpoint

$$1 - \left(\frac{\delta}{2K}\right)^{1/M},$$

while the interval for M successes has lower endpoint

$$\left(\frac{\delta}{2K}\right)^{1/M}.$$

These are the most separated confidence intervals among all possible success counts. Therefore, nontrivial reweighting is possible only if

$$\left(\frac{\delta}{2K}\right)^{1/M} > 1 - \left(\frac{\delta}{2K}\right)^{1/M},$$

or equivalently,

$$M > \log_2\left(\frac{2K}{\delta}\right).$$

For $K = 4$ and $\delta = 0.05$, this threshold is $\log_2(160) \approx 7.32$, so activation is impossible at $M = 2$ and $M = 4$ and first becomes possible at $M = 8$. At $M = 8$, the intervals for zero and eight successes are approximately $[0, 0.47]$ and $[0.53, 1]$, while every other pair of success-count intervals overlaps. Hence, under this construction, TRACEBAND activates at $M = 8$ exactly when the pool contains both an all-failure and an all-success prefix. These activation statements follow from Eq. (13). For fixed binary outcomes, the common strictly increasing transformation in Eq. (9) preserves interval intersections, so changing $\beta > 0$ does not change activation. It can still change nontrivial weights and continuation targets. When activation is impossible, the prefix target is unchanged but continuation fitting need not be inactive.

G ADDITIONAL SAMPLED-PREFIX TARGET CONTROLS

We introduce five additional controls to distinguish the effect of the prefix-weight construction used by TRACEBAND from simpler shrinkage, confidence-based, or magnitude-matched alternatives. For a fixed sampled pool, all target controls use the same prefixes and continuation rewards. In end-to-end training, each method samples on-policy from its own evolving policy; the comparisons therefore match the training protocol and sampling budget, rather than the realized trajectories. The five additional controls are constructed from the plug-in weights $\widehat{w}_i = K\widehat{Z}_i / \sum_j \widehat{Z}_j$, with $\widehat{Z}_i$ from Eq. (6). The uncertainty set $\mathcal{C} = \prod_i [L_i, U_i]$ is the same log-partition box used by TRACEBAND. For the controls gated by the worst-case gain, we use

$$B(w) = \min_{\widetilde{Z}:\, \log \widetilde{Z} \in \mathcal{C}} \Delta\bar{\mathcal{J}}(w; \widetilde{Z}),$$

exactly as in Eq. (10).

Continuation-only. This control sets $w_i = 1$ for every sampled prefix and uses the same continuation targets $a_{i,m}$ defined in Eq. (15). We minimize the full projection loss in Eq. (17), including the prefix fitting term, with joint target masses $P_{i,m} = a_{i,m}/K$. Thus, the prefix target remains uniform while the continuation targets retain their reward-dependent weights.

Fixed Shrinkage. Fixed Shrinkage retains the plug-in reweighting direction but reduces its magnitude using a fixed coefficient $\lambda \in [0, 1]$:

$$w^{\mathrm{FS}}(\lambda) = \mathbf{1} + \lambda(\widehat{w} - \mathbf{1}).$$

Thus, $\lambda = 0$ recovers the unchanged prefix target, whereas $\lambda = 1$ recovers the full plug-in target. Unlike TRACEBAND, the shrinkage coefficient is fixed and does not adapt to the uncertainty of the sampled prefixes. We use $\lambda = 0.5$ for the fixed-shrinkage control.

Worst-case Interpolation. Worst-case Interpolation keeps the plug-in reweighting direction but chooses how far to move from the unchanged target toward $\widehat{w}$. We define

$$w^{\mathrm{WI}} = \mathbf{1} + \alpha^{\mathrm{WI}}(\widehat{w} - \mathbf{1}), \qquad \alpha^{\mathrm{WI}} \in \arg\max_{\alpha \in [0,1]} B(\mathbf{1} + \alpha(\widehat{w} - \mathbf{1})) .$$

Worst-case Interpolation guarantees nonnegative true pool improvement on the simultaneous coverage event, since the feasible set always contains $\alpha = 0$, which recovers the no-change solution.

Matched Plug-in. Matched Plug-in uses the same prompt-level activation pattern as TRACEBAND, while retaining the plug-in reweighting direction. When $w^{\mathrm{TB}} = \mathbf{1}$, we set $w^{\mathrm{MP}} = \mathbf{1}$. Otherwise,

$$w^{\mathrm{MP}} = \mathbf{1} + \alpha^{\mathrm{MP}}(\widehat{w} - \mathbf{1}).$$

We match the update magnitude using the sampled-prefix KL

$$D_{\mathrm{pre}}(w) = D_{\mathrm{KL}}(p(w) \| p^0) = \frac{1}{K} \sum_{i=1}^{K} w_i \log w_i.$$

For activated prompts, we choose

$$\alpha^{\mathrm{MP}} \in \arg\min_{\alpha \in [0,1]} \left| D_{\mathrm{pre}}\left(\mathbf{1} + \alpha(\widehat{w} - \mathbf{1})\right) - D_{\mathrm{pre}}\left(w^{\mathrm{TB}}\right) \right| .$$

The constraint $\alpha \in [0, 1]$ clips any magnitude match that would require $\alpha > 1$ at the full plug-in update. Thus this is the closest attainable KL match along the plug-in segment, not necessarily an exact match. Unlike the controls that enforce a nonnegative worst-case gain, Matched Plug-in is a magnitude control and does not itself enforce $B(w) \geq 0$.

Random-gated Plug-in. Random-gated Plug-in matches the number of prompt-level pools activated by TRACEBAND within each rollout-update iteration, but selects those pools uniformly at random. It applies the Plug-in update to the selected pools and leaves the remaining pools unchanged. This control isolates whether update sparsity alone explains the gain from confidence-based activation.

H CONTROLLED VALIDATION WITH KNOWN LOG-PARTITION FUNCTIONS

We evaluate the finite-sample target construction in a controlled setting where known success probabilities determine the prefix partition functions, and hence the log-partition functions, exactly. For prefix i,

$$R \sim \mathrm{Bernoulli}(\rho_i), \qquad Z_i = 1 + (e^{\beta} - 1)\rho_i.$$

The oracle weights are

$$w_i^{\star} = \frac{K Z_i}{\sum_j Z_j},$$

and the true sampled-prefix improvement relative to $w = \mathbf{1}$ is

$$\Delta^{\star}(w) := \Delta\bar{\mathcal{J}}(w; Z) = \frac{1}{\beta K} \sum_{i=1}^{K} \left[(w_i - 1) \log Z_i - w_i \log w_i \right] .$$

The target rules observe only M sampled binary rewards per prefix. For $K = 4$, we enumerate all $(k_1, \ldots, k_4) \in \{0, \ldots, M\}^4$ and weight each outcome by its exact binomial probability, so the reported results contain no Monte Carlo error.

Table 4: Exact known-truth evaluation at $K = 4$, $M = 8$, $\beta = 1$, and $\delta = 0.05$. Coverage is simultaneous across all four prefixes.

Setting	Method	Simultaneous Coverage (%)	Activation Rate (%)	Harmful Update Rate (%)	Mean True Improvement	Oracle Regret
Equal	Plug-in	96.91	98.95	98.95	-1.03×10^{-2}	1.03×10^{-2}
	Worst-case Interpolation		1.67×10^{-3}	1.67×10^{-3}	≈ 0	≈ 0
	TraceBand		1.82×10^{-2}	1.82×10^{-2}	≈ 0	≈ 0
Weak	Plug-in	99.53	99.30	84.22	-7.42×10^{-3}	1.01×10^{-2}
	Worst-case Interpolation		5.77×10^{-3}	1.22×10^{-4}	≈ 0	2.68×10^{-3}
	TraceBand		6.98×10^{-2}	1.62×10^{-3}	5.52×10^{-7}	2.68×10^{-3}
Rare	Plug-in	98.76	95.42	55.75	-2.15×10^{-3}	7.11×10^{-3}
	Worst-case Interpolation		1.90×10^{-4}	$< 10^{-6}$	≈ 0	4.96×10^{-3}
	TraceBand		2.50×10^{-4}	$< 10^{-6}$	≈ 0	4.96×10^{-3}
Strong	Plug-in	98.74	100.00	4.26×10^{-2}	3.88×10^{-2}	6.60×10^{-3}
	Worst-case Interpolation		0.756	$< 10^{-6}$	2.84×10^{-5}	4.54×10^{-2}
	TraceBand		21.47	$< 10^{-6}$	1.10×10^{-3}	4.43×10^{-2}

We consider the following success-probability configurations:

$$\text{Equal: } (0.50, 0.50, 0.50, 0.50), \quad \text{Weak: } (0.40, 0.45, 0.55, 0.60),$$
$$\text{Rare: } (0.02, 0.05, 0.10, 0.20), \quad \text{Strong: } (0.10, 0.30, 0.70, 0.90),$$

These cover equal, weakly separated, low-success, and strongly separated regimes of the underlying success probabilities and log-partition functions. We compare the target rules defined in Appendix G. Activation Rate is the probability that $w \neq \mathbf{1}$, and Harmful Update Rate is the probability that $\Delta^\star(w) < 0$. Oracle Regret is $\Delta\bar{\mathcal{J}}(w^\star; Z) - \Delta\bar{\mathcal{J}}(w; Z)$, always evaluated on the same pool using the known Z_i. These are target-level quantities, not performance measurements after language-model fitting.

Table 4 shows that direct point-estimate reweighting can be harmful when finite samples do not clearly distinguish the log-partition functions, while Plug-in performs well once these functions are strongly separated. The small nonzero harmful-update rate of TRACEBAND occurs only when the simultaneous confidence intervals fail to cover the true values.

We next vary the continuation budget $M \in \{4, 8, 16, 32\}$ at $\delta = 0.05$, and the confidence parameter $\delta \in \{0.01, 0.05, 0.10, 0.20\}$ at $M = 8$. We use the Strong configuration throughout, with $K = 4$ and $\beta = 1$. Table 5 illustrates the trade-off between conservative target changes and available improvement. With very limited evidence, TraceBand leaves the prefix target unchanged; as the continuation budget increases, its mean true improvement increases. In this Strong configuration, Plug-in has higher mean improvement at every reported budget, whereas the rules gated by the worst-case gain reduce harmful-update probability at the cost of a more conservative reweighting. TraceBand can use better-supported differences without being restricted to the full plug-in update or its direction. The δ sweep changes both the allowed activation patterns and the magnitude of active weights; different confidence levels can share the same activation pattern. These controlled results evaluate the prefix target alone. They do not establish improvement of the fitted joint policy or measure the contribution of continuation learning.

I ADDITIONAL EXPERIMENTS

I.1 ADDITIONAL PREFIX-UPDATE ABLATIONS

We provide additional matched-pipeline comparisons of prefix-update controls. Table 6 reports benchmark-wise results for $K = 4$ at both the default setting $M = 8$ and the higher continuation budget $M = 16$, where tighter confidence bands permit nontrivial reweighting of mixed-success prefixes.

Table 5: Exact sensitivity analysis in the Strong configuration $(0.10, 0.30, 0.70, 0.90)$.

A. Continuation budget

M	Method	Activation Rate (%)	Harmful Update Rate (%)	Mean True Improvement
4	Plug-in	99.98	2.00	3.22×10^{-2}
	Worst-case Interpolation	0	0	0
	TraceBand	0	0	0
8	Plug-in	100.00	4.26×10^{-2}	3.88×10^{-2}
	Worst-case Interpolation	0.756	$< 10^{-6}$	2.84×10^{-5}
	TraceBand	21.47	$< 10^{-6}$	1.10×10^{-3}
16	Plug-in	100.00	2.63×10^{-5}	4.21×10^{-2}
	Worst-case Interpolation	72.82	$< 10^{-6}$	9.34×10^{-3}
	TraceBand	92.84	$< 10^{-6}$	1.43×10^{-2}
32	Plug-in	100.00	$< 10^{-6}$	4.38×10^{-2}
	Worst-case Interpolation	99.96	$< 10^{-6}$	2.75×10^{-2}
	TraceBand	100.00	$< 10^{-6}$	2.80×10^{-2}

B. Confidence parameter

δ	Method	Activation Rate (%)	Harmful Update Rate (%)	Mean True Improvement
0.01	Plug-in	100.00	4.26×10^{-2}	3.88×10^{-2}
	Worst-case Interpolation	0	0	0
	TraceBand	0	0	0
0.05	Plug-in	100.00	4.26×10^{-2}	3.88×10^{-2}
	Worst-case Interpolation	0.756	$< 10^{-6}$	2.84×10^{-5}
	TraceBand	21.47	$< 10^{-6}$	1.10×10^{-3}
0.10	Plug-in	100.00	4.26×10^{-2}	3.88×10^{-2}
	Worst-case Interpolation	14.29	$< 10^{-6}$	6.32×10^{-4}
	TraceBand	58.32	8.29×10^{-6}	2.75×10^{-3}
0.20	Plug-in	100.00	4.26×10^{-2}	3.88×10^{-2}
	Worst-case Interpolation	38.03	1.74×10^{-6}	3.47×10^{-3}
	TraceBand	58.32	8.30×10^{-6}	7.33×10^{-3}

Table 6: Matched-pipeline comparison of prefix-update rules with $K = 4$ under (a) $M = 8$ and (b) $M = 16$. Results are Pass@1 mean $\pm$ standard deviation over three independent training runs, with each trained model evaluated using deterministic greedy decoding. **Bold** indicates the best result.

Method	**MATH500**	**AMC23**	**CollegeMath**	**MinervaMath**	**AIME24**	**AIME25**
(a) $K = 4,\ M = 8$						
Continuation-only	87.9 ± 0.8	62.5 ± 2.5	28.3 ± 0.4	33.7 ± 0.2	18.9 ± 5.1	16.7 ± 3.3
Plug-in	88.1 ± 0.5	63.3 ± 3.8	29.2 ± 0.4	34.4 ± 1.2	21.1 ± 3.8	16.7 ± 3.3
Shrinkage ($\lambda = 0.50$)	88.5 ± 0.8	65.0 ± 2.5	29.5 ± 0.5	35.0 ± 0.6	22.2 ± 3.8	17.8 ± 5.1
Worst-case Interpolation	88.9 ± 0.9	65.0 ± 2.5	29.7 ± 0.2	35.7 ± 1.0	21.1 ± 5.1	16.7 ± 3.3
Matched Plug-in	88.7 ± 0.6	65.0 ± 2.5	29.8 ± 0.5	35.0 ± 0.8	22.2 ± 7.7	16.7 ± 5.8
Random-gated Plug-in	87.6 ± 0.5	63.3 ± 3.8	28.7 ± 0.4	33.9 ± 0.9	18.9 ± 3.8	15.6 ± 6.9
TRACEBAND	$\mathbf{89.7 \pm 0.6}$	$\mathbf{67.5 \pm 2.5}$	$\mathbf{30.7 \pm 0.3}$	$\mathbf{36.8 \pm 0.4}$	$\mathbf{25.6 \pm 1.9}$	$\mathbf{18.9 \pm 1.9}$
(b) $K = 4,\ M = 16$						
Continuation-only	88.1 ± 0.8	64.2 ± 2.9	29.3 ± 0.5	34.8 ± 0.6	21.1 ± 1.9	16.7 ± 3.3
Plug-in	88.9 ± 0.8	66.7 ± 2.9	29.8 ± 0.5	35.5 ± 0.6	23.3 ± 3.3	18.9 ± 5.1
Shrinkage ($\lambda = 0.50$)	88.9 ± 0.6	66.7 ± 5.2	30.0 ± 0.4	35.3 ± 1.0	23.3 ± 3.3	18.9 ± 3.8
Worst-case Interpolation	89.1 ± 0.8	$\mathbf{67.5 \pm 2.5}$	30.3 ± 0.6	35.8 ± 1.1	23.3 ± 3.3	$\mathbf{20.0 \pm 3.3}$
Matched Plug-in	89.1 ± 0.9	66.7 ± 2.9	30.4 ± 0.2	35.7 ± 0.4	24.4 ± 5.1	18.9 ± 3.8
Random-gated Plug-in	88.3 ± 0.6	65.0 ± 2.5	29.5 ± 0.5	34.8 ± 1.5	21.1 ± 6.9	17.8 ± 5.1
TRACEBAND	$\mathbf{89.9 \pm 0.3}$	$\mathbf{67.5 \pm 2.5}$	$\mathbf{31.0 \pm 0.2}$	$\mathbf{36.9 \pm 0.2}$	$\mathbf{25.6 \pm 1.9}$	$\mathbf{20.0 \pm 3.3}$

I.2 Observed Success Counts and Prefix Weights at M=8

We examine the prefix pools collected during the three independent TRACEBAND training runs used in the main mathematical-reasoning experiments. For prefix i in a prompt-level pool, let

$$c_i = \sum_{m=1}^{8} R_{i,m} \in \{0, \dots, 8\}$$

denote the number of successful continuations among the $M = 8$ samples. Under $K = 4$ and $\delta = 0.05$, Eq. (13) implies

$$w^{\mathrm{TB}} \neq \mathbf{1} \quad \Longleftrightarrow \quad \min_i c_i = 0 \text{ and } \max_i c_i = 8.$$

For frequency reporting only, we sort the four success counts as $c_{(1)} \leq c_{(2)} \leq c_{(3)} \leq c_{(4)}$; the original prefix indices are retained when constructing the training target. Table 7 reports representative success-count tuples together with the corresponding TRACEBAND and Matched Plug-in weights.

The observed tuples illustrate that matching the reweighted pools and the magnitude of the prefix reweighting does not generally recover the same weights. For many active pools, TRACEBAND leaves intermediate prefixes at their baseline weight because their confidence intervals contain the threshold, whereas Matched Plug-in also changes these weights as it remains constrained to the Plug-in direction. Across all reweighted pools, the two prefix-weight vectors differ in 82.4% of cases.

Table 7: Observed prefix weights among the 25,536 pools with non-uniform TRACEBAND weights under $K = 4$, $M = 8$, and $\delta = 0.05$. Fractions for individual tuples are computed over these reweighted pools.

Success counts	**Fraction (%)**	$\mathbf{w}^{\mathrm{TB}}$	$\mathbf{w}^{\mathrm{MP}}$	$\lVert \mathbf{w}^{\mathrm{TB}} - \mathbf{w}^{\mathrm{MP}} \rVert_\infty$
Overall: $\mathbf{w}^{\mathrm{TB}} \neq \mathbf{w}^{\mathrm{MP}}$ in $\mathbf{82.4}\%$ of reweighted pools				
Representative observed success-count tuples				
$(0, 4, 4, 8)$	10.11	$(0.9720, 1.0000, 1.0000, 1.0280)$	$(0.9720, 1.0000, 1.0000, 1.0280)$	0.0000
$(0, 0, 8, 8)$	3.89	$(0.9720, 0.9720, 1.0280, 1.0280)$	$(0.9720, 0.9720, 1.0280, 1.0280)$	0.0000
$(0, 3, 5, 8)$	3.69	$(0.9720, 1.0000, 1.0000, 1.0280)$	$(0.9729, 0.9932, 1.0068, 1.0271)$	0.0068
$(0, 3, 4, 8)$	3.53	$(0.9720, 1.0000, 1.0000, 1.0280)$	$(0.9741, 0.9948, 1.0017, 1.0294)$	0.0052
$(0, 4, 5, 8)$	3.65	$(0.9720, 1.0000, 1.0000, 1.0280)$	$(0.9707, 0.9983, 1.0052, 1.0259)$	0.0052
$(0, 2, 5, 8)$	3.37	$(0.9720, 1.0000, 1.0000, 1.0280)$	$(0.9755, 0.9886, 1.0082, 1.0277)$	0.0114
$(0, 3, 6, 8)$	3.31	$(0.9720, 1.0000, 1.0000, 1.0280)$	$(0.9723, 0.9919, 1.0114, 1.0244)$	0.0114
$(0, 2, 6, 8)$	3.01	$(0.9720, 1.0000, 1.0000, 1.0280)$	$(0.9750, 0.9875, 1.0125, 1.0250)$	0.0125

I.3 Independent reference diagnostic

For the reference-based diagnostics, we draw $M_{\mathrm{ref}} = 256$ independent continuations from each sampled prefix and construct two-sided Clopper–Pearson intervals with $\delta_{\mathrm{ref}} = 0.05$, with δ_{ref}/K for each prefix and $\delta_{\mathrm{ref}}/(2K)$ for each tail. We map these intervals to simultaneous reference intervals $[L_i^{\mathrm{ref}}, U_i^{\mathrm{ref}}]$ for $\log Z_i$. For any fixed prefix weights w, the lower improvement bound uses L_i^{ref} when $w_i \geq 1$ and U_i^{ref} otherwise, while the upper bound reverses these choices.

I.4 Projection Sign Analysis

We evaluate whether the sampled-prefix improvement remains positive after projection to the parameterized language model using the independent reference diagnostic in Appendix I.3. We randomly sample 1,000 active and 1,000 inactive prompt-level prefix pools from three independent training runs, with no prompt appearing more than once.

Let $w_i^{\mathrm{proj}} = K q_i^{\mathrm{pre}}(\theta)$ denote the prefix weights induced by the fitted policy. Applying the independent reference diagnostic to w^{proj}, we classify a case as positive if its reference lower bound is above zero, negative if its reference upper bound is below zero, and unresolved otherwise.

Table 8: Sign of the projected sampled-prefix improvement under the independent reference diagnostic ($M_{\mathrm{ref}} = 256$).

Case	Positive	Unresolved	Negative
Active	92.10% (921)	7.70% (77)	0.20% (2)
Inactive	4.7% (47)	89.6% (896)	5.7% (57)

Each row contains $N = 1{,}000$ prompt-level prefix pools.

Overall, the diagnostic suggests that TRACEBAND's positive update direction is largely preserved after projection, while inactive cases show no consistent directional change.

I.5 TRAINING OBJECTIVE ABLATION

We test whether the advantage of TRACEBAND depends on the fitting objective. As an alternative to the default ratio-based projection, we fit the same prefix and continuation targets with a weighted log-likelihood objective. For the chosen prefix target w (either $\widehat{w}$ for Plug-in or w^{TB} for TRACEBAND), we use

$$\mathcal{L}_{\mathrm{WLL}}(\theta; x) = -\frac{1}{K}\sum_{i=1}^{K} w_i \left[\log \pi_\theta(s_i \mid x) + \sum_{m=1}^{M} a_{i,m} \log \pi_\theta(c_{i,m} \mid x, s_i)\right].$$

Table 9: Average Pass@1 under alternative fitting objectives.

Prefix target	Ratio projection	Weighted log-likelihood
Plug-in	42.1	41.9
TRACEBAND	**44.9**	**43.5**

TRACEBAND outperforms the corresponding Plug-in target under both fitting objectives, indicating that its advantage is not specific to the default ratio-based projection.

I.6 TRAINING COST

We compare GRPO, PPPO, TEMPO, and TRACEBAND in the default mathematical-reasoning setting with Qwen2.5-Math-7B. All methods use the same verifier budget of 32 reward-evaluated completions per prompt and are trained on 4 NVIDIA H200 GPUs. TRACEBAND additionally generates $K = 4$ non-reward-evaluated initial full rollouts for prefix construction; these rollouts, including discarded suffixes, are included in the token accounting.

As shown in Table 10, TRACEBAND uses fewer generated tokens than all three baselines despite the additional initial rollouts, because its fresh continuations begin from intermediate prefixes and are shorter on average. Its two-stage generation and projection procedure incurs only modest wall-clock overhead. Since all methods use the same verifier budget, the performance gains cannot be attributed to additional reward evaluations. High-budget reference samples used only for diagnostics are excluded from these measurements.

Table 10: Training cost for the default mathematical-reasoning setting on Qwen2.5-Math-7B. All methods use the same verifier budget of 32 reward-evaluated completions per prompt and are trained on 4 NVIDIA H200 GPUs. For TRACEBAND, token counts include the four initial full rollouts used for prefix construction.

Method	Generated Tokens/Prompt	Wall-clock Time
GRPO	60.8k	27.4h
PPPO	49.4k	27.7h
TEMPO	58.8k	28.3h
TRACEBAND	42.6k	28.1h

I.7 CUTOFF ABLATION

For each prompt, let $T_{\min}$ denote the length of the shortest of the K initial rollouts. Early, Middle, and Late set the shared cutoff to $\lfloor 0.25T_{\min} \rfloor$, $\lfloor 0.50T_{\min} \rfloor$, and $\lfloor 0.75T_{\min} \rfloor$, respectively, subject to the same minimum 16-token prefix and continuation constraints as the default setting. Random samples the shared cutoff from the feasible positions as in the default setting. All settings use matched Continuation-only and Plug-in controls.

Table 11: Effect of prefix location under matched sampling and training budgets.

Cutoff	Continuation-only	Plug-in	TraceBand	Activation (%)
Early	41.1	41.3	41.5	3.8
Middle	41.3	42.0	43.8	11.6
Late	41.8	42.4	43.0	21.4
Random	41.3	42.1	44.9	13.3

Table 11 shows that TRACEBAND outperforms its matched controls at every cutoff. Its incremental gain is largest at Middle and Random, whereas Late activates most frequently but yields a smaller gain over Plug-in. Thus, the reported performance ordering is not explained by activation frequency alone. These comparisons do not by themselves determine whether a retained prefix already contains a final answer. Random performs best overall among the tested cutoff rules.

I.8 HYPERPARAMETER SENSITIVITY

Table 12 reports sensitivity to the confidence parameter δ and KL-regularization parameter β.

Confidence parameter δ. We vary $\delta \in \{0.01, 0.05, 0.10, 0.20\}$ with $K = 4$ and $M = 8$. At $\delta = 0.01$, the wider confidence intervals prevent prefix reweighting. For fixed success counts, $\delta = 0.05$ permits activation only when counts 0 and 8 both occur in the pool. At either $\delta = 0.10$ or 0.20, the disjoint count pairs are $(0, 7)$, $(0, 8)$, and $(1, 8)$. Thus the latter two settings have identical gates on identical outcomes, although their weights differ. Separately trained runs also include changes in the on-policy samples. Among the tested values, $\delta = 0.05$ achieves the highest average Pass@1.

KL-regularization parameter β. We vary $\beta \in \{0.25, 0.5, 1, 2\}$ with the remaining hyperparameters fixed. Average Pass@1 is highest at $\beta = 1$, while activation remains similar across the sweep (13.1%–13.7%). For fixed continuation outcomes, changing β preserves the activation decision but changes the prefix weights and continuation targets. The small variation in aggregate activation reflects differences in the on-policy samples generated by the separately trained models.

Table 12: Sensitivity to the confidence parameter δ and KL-regularization parameter β.

	Confidence Parameter δ				KL Regularization β			
Metric	0.01	0.05	0.10	0.20	0.25	0.50	1.00	2.00
Avg. Pass@1 ↑	41.3	44.9	43.4	43.0	42.9	43.4	44.9	43.3
Activation Rate (%)	0.0	13.3	21.1	21.1	13.1	13.5	13.3	13.7

J IMPLEMENTATION DETAILS

J.1 TRAINING CONFIGURATION

Table 13 summarizes the training hyperparameters used for mathematical reasoning and code generation. Within each task, shared training and sampling hyperparameters are kept fixed across all methods unless otherwise specified.

Table 13: Training hyperparameters for mathematical reasoning and code generation.

Hyperparameter	**Math Reasoning**	**Code Generation**
Optimizer	AdamW	AdamW
Learning rate	1×10^{-6}	5×10^{-5}
LR scheduler	Cosine	Cosine
Warmup ratio	0.0	0.0
Per-device prompt batch size	16	8
Gradient accumulation steps	8	8
Rollout-update iterations	1000	1000
Optimizer updates per iteration	3	3
Maximum prompt length	1024	2048
Maximum response length	2048	2048
Rollout temperature	1.0	1.0
Rollout top-p	1.0	1.0
Precision	BF16	BF16
Number of GPUs	4	4
GPU type	NVIDIA H200	NVIDIA H200
PEFT	LoRA	LoRA
LoRA target modules	All linear layers	All linear layers
LoRA rank / alpha	32 / 32	32 / 32
LoRA dropout	0.0	0.0
Cutoff mode	Shared random	Shared random
Minimum prefix length	16	16
Minimum continuation length	16	16
Number of prefixes K	4	4
Continuations per prefix M	8	8
Confidence parameter δ	0.05	0.05
KL parameter β	1.0	1.0

Mathematical-reasoning verifier. For mathematical reasoning, the verifier extracts the final answer from the last `\boxed{}` expression in the generated response and compares it with the reference using normalized answer matching, including numeric and symbolic equivalence checks. The same verifier is used across all methods.

Code-generation setup. For code generation, we train on the TACO training split after excluding image-dependent problems and examples without usable reference solutions. The execution-based verifier uses up to 32 deterministic tests per problem. We evaluate on HumanEval+, MBPP+, LiveCodeBench `release_v4` (713 problems), and the full BigCodeBench-Complete split (1,140 tasks) using their respective execution harnesses. Each trained model generates one completion per problem using deterministic greedy decoding. We report the mean and standard deviation of Pass@1 across three independent training runs.

J.2 Prefix and Continuation Sampling

For each training prompt, we sample K independent rollouts from π_{old}. We identify the shortest rollout and sample a common cutoff from its valid token positions, then truncate all K rollouts at that cutoff to obtain prefixes $s_1, \ldots, s_K$. This compares all prefixes at the same generation depth without discarding sampled rollouts.

From each prefix s_i, we draw M fresh continuations from $\pi_{\text{old}}(\cdot \mid x, s_i)$. All generations are subject to a 2048-token cap on the complete generated response, including the retained prefix; hence, a prefix of length $|s_i|$ leaves at most $2048 - |s_i|$ tokens for its continuation. Continuation inputs are formed by directly concatenating the prompt and sampled prefix token IDs, avoiding decoding and re-tokenization.

Both the initial rollouts and fresh continuations are sampled from the raw π_{old} distribution with temperature 1 and without top-p or top-k filtering. Sequence log probabilities used in Eq. (16) are computed by summing token-level log probabilities under the same distribution.

The finite-sample confidence guarantees are interpreted conditional on the realized prefix pool and common cutoff. One rollout-update iteration collects a fresh batch of prompts and evaluates their rewards once, followed by three optimizer updates on the fixed sampled pools and targets. The per-device prompt batch size denotes the number of fresh prompts collected on each device in one rollout-update iteration. Gradient accumulation is applied only within optimization and does not change the number of fresh prompts or reward evaluations. The cosine learning-rate scheduler is advanced at each optimizer update. For TRACEBAND, the sampled pools, prefix and continuation targets, and π_{old} remain fixed across these three updates; afterward, π_{old} is refreshed and a new batch is collected. Thus, 1,000 rollout-update iterations correspond to 3,000 optimizer updates.

J.3 BASELINE IMPLEMENTATION AND EVALUATION-BUDGET MATCHING

For all methods, we use the same base model, training dataset, maximum response length, number of rollout-update iterations, and total number of fresh training prompts within each task. We further match the verifier budget to 32 fresh reward evaluations per prompt. Each collected pool is reused for three optimizer updates for all methods, without additional reward evaluations.

We preserve each baseline's method-specific credit-assignment mechanism while adapting its rollout scheme when necessary to satisfy this common budget. For DAPO, we disable dynamic sampling, which would otherwise alter the sampling budget, while retaining the remaining optimization procedure. For VinePPO, we use two reward-evaluated training trajectories per prompt, uniformly sample two valid intermediate reasoning states from each trajectory, and draw six Monte Carlo continuations from each of the four selected states and six from the shared initial state. Including the two original trajectories, this gives $2 + 4 \times 6 + 6 = 32$ reward evaluations per prompt. For PPPO, we use four initial outputs and seven additional continuations per retained prefix; counting the original output gives $4 \times (7 + 1) = 32$ reward evaluations. TEMPO constructs its prefix tree from the same 32 reward-evaluated responses without additional verifier calls. For GRPO-MA, we retain its native post-reasoning breakpoint and use four thoughts with eight answer continuations per thought, giving $4 \times 8 = 32$ reward evaluations.

For TRACEBAND, the 32 verifier calls correspond to $K = 4$ prefixes with $M = 8$ fresh continuations per prefix. The K initial full rollouts used to construct the shared cutoff are not reward-evaluated and therefore do not enter the verifier budget, but their full generation cost, including discarded suffixes, is included in our compute accounting. Generated-token and wall-clock costs are reported separately in Appendix I.6.