RATIO-FREE POLICY OPTIMIZATION: A BIAS-VARIANCE VIEW OF ON-POLICY FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Practical LLM reinforcement learning is never strictly on-policy: each batch of rollouts is reused for many optimizer steps, and rollouts come from a separate inference engine. We show that GRPO degrades under both drifts — and on coding collapses — and explain why. We formalize *on-policy fine-tuning (OPFT)*, updating a model on its own rollouts, as the *OPFT loop*, of which GRPO and iterative rejection-sampling SFT are realizations, and propose RFPO (Ratio-Free Policy Optimization), a novel realization that optimizes the surrogate $J_{\mathrm{RF}}(\pi_\theta) = \mathbb{E}_{\pi_{\text{old}}}[A \log \pi_\theta]$ — a reward-weighted maximum-likelihood objective requiring no importance ratio — as a local proxy for the on-policy expected reward $J(\pi_\theta)$. We identify three mechanisms — z-score advantages, negative samples, and trust-region clipping — that make this surrogate stable, each with a distinct role verified by theory and ablations; the analysis also explains the well-known collapse of iterative rejection-sampling SFT. GRPO realizes analogous mechanisms through different machinery: the core updates of RFPO and GRPO coincide at strict on-policy and diverge under the practical sampling gaps that any modern LLM RL stack faces (multiple optimizer steps per batch; decoupled rollout/training engines). A bias-variance analysis predicts that GRPO's importance-sampling variance scales with the gap while RFPO's bias is controlled by a per-token trust region, so RFPO dominates in MSE under sufficient drift. On Qwen3-4B training on POLARIS/DeepCoder we verify the prediction along both drift axes: as we increase optimizer steps per batch, GRPO degrades from 0.90 (MATH-500)/0.67 (LCBv6) at 1 step through 0.87/0.38 at 8 steps to 0.85/0.24 at 32 steps, while RFPO stays at 0.90/0.67; under bypass-mode rollout drift (8 steps), GRPO collapses ($0.87/0.38 \rightarrow 0.74/0.04$) while RFPO is nearly unaffected (0.90/0.67) — and, needing no reference model, KL term or importance ratio, RFPO trains 1.3–1.9$\times$ faster per step.

1 INTRODUCTION

Pre-training and supervised fine-tuning (SFT) fit a model by maximum likelihood to a fixed external corpus; reinforcement learning (RL) post-training instead trains the model on its *own* rollouts, scored by a reward, so the training distribution shifts as the policy updates. This on-policy property is what makes RL effective — the gradient points where the current policy can improve — and RL is often reported to forget less than continued SFT (Shenfeld et al., 2025). GRPO (Shao et al., 2024), an actor-only group-relative policy-gradient recipe, has driven much of the recent progress in LLM reasoning.

In this paper we formalize this practice — training on the model's own rollouts — as the *on-policy fine-tuning (OPFT)* loop, an abstract algorithm in which generation-and-scoring alternates with mini-batch updates (Algorithm 1 in Section 3). GRPO is one realization of the OPFT loop; the natural maximum-likelihood realization, iterative rejection-sampling SFT (Gulcehre et al., 2023; Zelikman et al., 2022; Yuan et al., 2023), is another. We ask two questions. First, what is the right *objective* for the update step inside the OPFT loop — and is policy gradient with importance sampling actually necessary? Second, given an objective, what *mechanisms* are required to keep the OPFT loop stable? Our proposed answer is RFPO (Ratio-Free Policy Optimization), a recipe whose two contributions we summarize next.

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.

Strict on-policy is impractical. We use "on-policy" in the standard PPO/GRPO sense: rollouts come from the current policy and may be reused for several optimizer steps within a batch. The cleanest setting — rollout policy and actor identical at every gradient step — is inefficient and impractical for any modern LLM RL stack, for two distinct reasons:

1. **Policy drift (within-batch), from training efficiency.** Strict on-policy requires one optimizer step per training batch (mini-batch size equal to batch size). Generating thousands of rollouts to take a single gradient step wastes compute. Practical recipes use `mbs < bs`, taking $K = \texttt{bs}/\texttt{mbs}$ optimizer steps per batch (`mbs` counts prompts; this is sample reuse, not gradient accumulation, which keeps $K = 1$ and introduces no drift). The actor π_θ then drifts from the batch-start snapshot π_{old} across these steps; we call this deviation $\pi_\theta - \pi_{\text{old}}$ the *policy drift*, and it grows with the number of steps.
2. **Rollout drift (engine mismatch), from decoupled inference and training.** Modern frameworks separate the rollout engine (vLLM (Kwon et al., 2023), sglang) from the training engine (e.g., FSDP). Even at the same checkpoint, numerical mismatch is unavoidable: different attention kernels, dtype precision, possible quantization. So the rollout policy π^*_{old} — the distribution the samples were actually drawn from, realized by the inference engine — differs from the training-engine snapshot π_{old}. We call this deviation $\pi_{\text{old}} - \pi^*_{\text{old}}$ the *rollout drift*; unlike policy drift it is present even at "on-policy" and is static within a batch.

Policy drift and *rollout drift* are universal — every production realization of the OPFT loop faces them — and they are the two axes along which we compare RFPO and GRPO empirically (Section 5) and analyze them theoretically (Section 4.2). The actor's total deviation from the distribution its samples were drawn from decomposes as $\pi_\theta - \pi^*_{\text{old}} = (\pi_\theta - \pi_{\text{old}}) + (\pi_{\text{old}} - \pi^*_{\text{old}})$ — policy drift plus rollout drift. This is also why the bias-variance view is the right lens: at strict on-policy there is no drift to trade off, but in practice it is unavoidable, and methods differ in how they navigate it.

Contributions. We make three contributions.

(1) A bias-variance view of the OPFT loop. We frame GRPO and RFPO as two strategies for estimating the same on-policy gradient from off-policy rollouts: GRPO importance-corrects each token (less bias, but variance that grows with the drift gap $r_t = \pi_\theta/\pi_{\text{old}}$), while RFPO optimizes a surrogate J_{RF} whose definition needs no importance correction (a trust-region-controlled bias, but on-policy variance). The two coincide at strict on-policy and diverge under drift; a bias-variance analysis (Section 4.2) predicts the bounded-bias surrogate dominates in MSE once drift is large enough. In short: under drift, a controlled bias beats a growing variance.

(2) RFPO: a novel recipe for the OPFT loop with three mechanisms. We propose RFPO (Ratio-Free Policy Optimization), a concrete recipe that realizes a surrogate objective with three stabilizing mechanisms: *z-score advantages* (M1), *negative samples* via a centered advantage (M2), and a per-token *trust region* via absolute probability clipping (M3). A leave-one-out ablation shows M2 and M3 are necessary for stability and M1 for smooth optimization, and an audit of six alternative stabilizers (entropy, KL penalties, *etc.*) recovers none of the lost stability. Theoretically, the three play distinct roles — M1 balances per-prompt gradient contribution, M2 makes J_{RF} non-degenerate, M3 controls the bias of J_{RF} relative to J. GRPO realizes the same three mechanisms through different machinery (Section 4.3).

(3) A simpler, faster recipe. RFPO needs no reference model, KL term or importance ratio, trains 1.3–1.9$\times$ faster per step (Section 5.4), and changes only the update step, so it is a drop-in replacement for GRPO's update in existing pipelines.

We verify both contributions on Qwen3-4B across math (POLARIS-53K) and coding (DeepCoder-16K): RFPO matches strict-on-policy GRPO, holds under policy drift (smaller `mbs`) and is far more robust to rollout drift (bypass mode), where GRPO first degrades then collapses (Section 5).

2 Related Work

Policy-gradient and iterative-MLE realizations of the OPFT loop. The dominant realizations of the OPFT loop for LLM post-training are policy-gradient methods — PPO (Schulman et al., 2017) for RLHF (Ouyang et al., 2022), GRPO (Shao et al., 2024) which we compare against, DAPO (Yu et al.,

2025) — all of which optimize a clipped policy-gradient objective with importance sampling between π_θ and π_{old}. Critic-free REINFORCE-style estimators such as RLOO (Ahmadian et al., 2024) are ratio-free but assume fresh on-policy samples; RFPO's gradient has the same form but is applied to *reused* samples, and our contribution is the analysis of that reuse and the mechanisms that keep it stable. The natural maximum-likelihood counterpart is iterative rejection-sampling SFT (Gulcehre et al., 2023; Zelikman et al., 2022; Yuan et al., 2023), which is conceptually simpler but is widely reported to collapse iteratively (Singh et al., 2024); our analysis identifies the missing ingredient as the centered-advantage / negative-sample mechanism M2 of Section 4. DPO (Rafailov et al., 2023) provides a related contrastive signal, anchored to a reference model. Most directly comparable to our surrogate framing is MaxRL (Tajwar et al., 2026), which interpolates between standard RL and exact maximum-likelihood estimation through a compute-indexed family of sample-based objectives. Our work is complementary: rather than indexing by sampling compute, we identify the practical drift sources that any modern LLM RL stack faces (Section 1) and characterize the bias-variance trade-off along those axes.

Stability under rollout drift. Drift between the rollout policy and the actor is a first-class stability concern in modern LLM RL stacks. verl exposes a *bypass mode* that reuses vLLM rollout log-probabilities as the IS denominator, making the gap between the rollout policy π^*_{old} and the actor π_θ visible; we use this as an explicit drift axis (Section 5.2). Several concurrent works address the same drift along the orthogonal axis of *where to place the importance ratio / trust region within a sequence*. Zheng et al. (2025) propose GSPO, which replaces token-level importance ratios with a single sequence-level ratio and reports stabilization of MoE RL. Lei et al. (2026) propose MinPRO, which replaces the cumulative prefix ratio with the minimum token-level ratio observed in the preceding prefix. Li et al. (2025) derive sharper bounds on the maximum token-level divergence across a sequence and propose *trust-region masking* — masking entire sequences that violate the bound, with monotonic-improvement guarantees for long-horizon RL. All three replace the per-token clip with a sequence- or prefix-level trust region. RFPO sits at the per-token end of this design axis (M3 in Section 4); our claim is complementary — the surrogate J_{RF} and the bias-variance trade-off are independent of where the trust region is placed, and a sequence-level RFPO variant in the style of Zheng et al. (2025); Lei et al. (2026); Li et al. (2025) is left to future work (Section D). DAPO's non-ratio components (dynamic sampling, overlong shaping) are likewise complementary and combine with RFPO directly.

Bias-variance trade-offs in off-policy RL. Trading IS variance for surrogate bias is a long-standing pattern — a core motivation of trust-region methods (TRPO, PPO) and conservative policy iteration. Our contribution is to apply this lens to LLM on-policy fine-tuning explicitly: characterizing GRPO and RFPO as two corners of the trade-off, and verifying that the surrogate dominates under sufficient drift on both practical axes.

3 The OPFT loop and Its Realizations

We study the abstract OPFT loop in Algorithm 1: at each training batch, the policy generates and scores rollouts on a set of prompts, snapshots itself, and then performs $K = \texttt{bs}/\texttt{mbs}$ optimizer steps over mini-batches of the same rollout set before drawing the next batch. The OPFT loop is general enough to describe every method studied in this paper — realizations differ only in the implementation of UPDATE inside the inner mini-batch loop; the outer rollout-and-score loop is shared.

Static-reference methods train against a rollout distribution that does not respond to the model. Pretraining and SFT are the extreme case, lying outside the OPFT loop altogether: they minimize MLE on a fixed external corpus with no self-generated rollouts. Single-round rejection-sampling SFT (Gulcehre et al., 2023; Zelikman et al., 2022; Yuan et al., 2023) is instead the loop's *frozen boundary* — a single outer round ($\texttt{bs} \geq$ dataset size) in which the rollout policy π^*_{old} is held at the initialization — drawing rollouts once and fine-tuning on the filtered keepers. Both are stable because the training distribution does not respond to the model.

Drifting-reference methods re-snapshot the rollout policy each outer round ($\texttt{bs} <$ dataset size), so π_{old} tracks π_θ and the rollout distribution drifts with training. The natural maximum-likelihood realization, *iterative* RS-SFT, regenerates rollouts from the updated policy each round and trains MLE

Algorithm 1 The OPFT loop (abstract)

Require: Policy π_θ, prompts $\{x_i\}$, reward R, rollouts per prompt N, batch size `bs`, mini-batch size `mbs`, update routine UPDATE

for each training batch of size `bs` **do**
  $\pi_{\text{old}} \leftarrow \pi_\theta$ *(snapshot)*
  for each prompt x_i in the batch **do**
    $Y_i \leftarrow \{y_{i,1}, \ldots, y_{i,N}\} \sim \pi^*_{\text{old}}(\cdot \mid x_i)$ *(rollouts)*
    $r_{i,j} \leftarrow R(x_i, y_{i,j})$ *(scoring)*
  end for
  for each mini-batch m of `mbs` prompt groups in (X, Y, R) **do**
    $\pi_\theta \leftarrow \text{UPDATE}(\pi_\theta, \pi_{\text{old}}, m)$ *(design choice)*
  end for
end for

on the keepers — and is widely reported to collapse (Singh et al., 2024). GRPO (Shao et al., 2024) is the canonical stable realization, combining a clipped policy-gradient objective with importance sampling, group-relative z-score advantages, and a KL term against the initialization. Our recipe RFPO (Section 4) is a third drifting-reference realization.

Within these drifting-reference realizations, both drift axes of Section 1 live in the inner loop: the actor drifts from the batch-start snapshot ($\pi_\theta \neq \pi_{\text{old}}$, policy drift), and the inference-engine rollout policy differs from that snapshot even at batch start ($\pi^*_{\text{old}} \neq \pi_{\text{old}}$, rollout drift, exposed by verl's bypass mode). Here π_{old} is the training-engine snapshot used as the trust-region anchor and, for GRPO, the IS denominator.

Iterative RS-SFT is the one that collapses; Section 4.1 shows it is the *naive* RFPO recipe.

4 RFPO: A STABLE REALIZATION OF THE OPFT LOOP

This section develops RFPO, our recipe for the OPFT loop. We begin with the objective (Section 4.1), analyze it through the bias-variance lens against GRPO's importance-sampling alternative (Section 4.2), then introduce three mechanisms grounded in this analysis (Section 4.3); Section 4.4 states the full update routine.

4.1 A RATIO-FREE OBJECTIVE J_{RF}

Throughout this section and Section 4.2 we assume no rollout drift, i.e., $\pi^* = \pi$; we restore rollout drift at the end of Section 4.2. The on-policy expected reward $J(\pi_\theta) = \mathbb{E}_{y\sim\pi_\theta}[R(y)]$ is the quantity we ultimately want to maximize. The gradient of J is an expectation under the *current* policy π_θ while the rollouts are sampled under π_{old}. GRPO closes the gap with an importance-sampling (IS) correction $r_t = \pi_\theta(y_t|x, y_{<t})/\pi_{\text{old}}(y_t|x, y_{<t})$, yielding the IS surrogate $J_{\text{IS}}(\pi_\theta) = \mathbb{E}_{y\sim\pi_{\text{old}}}\big[\sum_t r_t\, A(y)\big]$ (before clipping), whose gradient is a per-token off-policy estimate of ∇J: it corrects each token's conditional but not the distribution of its prefix $y_{<t}$ or of the continuation that determines A, so it is exact only at strict on-policy. We propose instead to change the local target to the surrogate

$$J_{\text{RF}}(\pi_\theta) \;=\; \mathbb{E}_{y\sim\pi_{\text{old}}}\big[A(y)\log\pi_\theta(y)\big], \tag{1}$$

whose expectation is taken under the sampling distribution itself, so no importance ratio is required. At $\theta = \theta_{\text{old}}$, $\nabla J_{\text{RF}} = \nabla J$; a small step on J_{RF} improves J to first order. Due to autoregressive factorization, J_{RF}'s gradient decomposes into a sum of per-token score-function gradients, weighted by the sequence-level advantage $A(y)$. Similar to GRPO, RFPO operates on this token-level decomposition.

Connection to RS-SFT, and an MLE implementation. Iterative rejection-sampling SFT (Gulcehre et al., 2023; Zelikman et al., 2022; Yuan et al., 2023) is a special case of optimizing J_{RF} with the binary advantage $A(y) = \mathbf{1}[R(y) = 1]$, i.e., train MLE on the keepers, ignore the rest. More generally, J_{RF} is a reward-weighted maximum-likelihood objective in the family of reward- and advantage-weighted regression (Peters & Schaal, 2007; Peng et al., 2019); our contribution is not this objective but the ratio-free bias-variance analysis (Section 4.2) and the mechanisms that stabilize it

under drift. Because J_{RF} has no importance ratio, any choice of A admits an iterative weighted-MLE implementation: snapshot π_{old}, generate rollouts, compute A, and minimize $-\sum A(y) \log \pi_\theta(y)$ over mini-batches.

4.2 BIAS-VARIANCE TRADE-OFF

Both RFPO and GRPO target $\nabla J(\pi_\theta) = \mathbb{E}_{y\sim\pi_\theta}[A\nabla \log \pi_\theta(y)]$ from rollouts $y \sim \pi_{\text{old}}$ (no rollout drift, per Section 4.1; we restore it below). We compare their per-sample token gradient estimators $\hat{g}$ through $\text{MSE}(\hat{g}) = \text{Var}(\hat{g}) + \|\text{Bias}(\hat{g})\|^2$, the error of $\hat{g}$ as an estimate of ∇J. Off-policy samples force a choice in how to spend this error: correct the sampling mismatch with an importance ratio, *reducing the bias* but paying variance that grows with policy drift; or change the target so its expectation already matches the sampler, *bounding the variance* but paying bias. GRPO and RFPO sit at these two corners.

GRPO — correct the ratio. Reweighting by $r_t = \pi_\theta/\pi_{\text{old}}$ gives $\hat{g}_{\text{IS}} = r_t \cdot A \cdot \nabla \log \pi_\theta$, the per-sample gradient of J_{IS} (Section 4.1). Up to clipping, its bias is this residual mismatch — accepted to avoid the instability of the exact, cumulative prefix ratio (Lei et al., 2026) — so its dominant error is variance, which scales with $\mathbb{E}_{\pi_{\text{old}}}[r_t^2]$ and grows with policy drift. Ratio clipping bounds the per-token variance to $(1+\epsilon)^2\sigma^2_{\text{on-policy}}$ (at the cost of a clipping bias beyond the band), but the variance still compounds across the $K = \texttt{bs}/\texttt{mbs}$ optimizer steps in a batch: a token pushed up at step 1 has $r_t > 1$ at step 2, amplifying the next update, so the cumulative move is super-linear in K.

RFPO — bound the variance. RFPO instead optimizes the surrogate J_{RF} (1), whose expectation already matches the sampler, so its per-sample gradient $\hat{g}_{\text{RF}} = A \cdot \nabla \log \pi_\theta$ carries no ratio. It is therefore unbiased for ∇J_{RF} but biased for ∇J — the price of dropping the ratio. This bias is governed by the policy drift $\Delta(y) = \pi_\theta(y) - \pi_{\text{old}}(y)$:

$$\nabla J - \nabla J_{\text{RF}} = \textstyle\sum_y \Delta(y)\, A(y)\, \nabla \log \pi_\theta(y), \tag{2}$$

which is zero at strict on-policy and grows with the drift. M3 controls it *locally*, capping each token's drift in the advantage direction at δ (Section B). By contrast, its variance $\text{Var}(\hat{g}_{\text{RF}}) = \mathbb{E}_{\pi_{\text{old}}}\left[\|A\nabla \log \pi_\theta\|^2\right] - \|\nabla J_{\text{RF}}\|^2$ carries no r_t^2 inflation — it sits at the strict-on-policy floor independent of policy drift — so it stays bounded.

Under sufficient drift, GRPO's variance exceeds RFPO's bounded squared bias, so the surrogate attains the lower MSE for ∇J; the prediction is monotonic in drift and is tested along both axes in Section 5. See Appendix B for the full analysis.

Restoring rollout drift. We have assumed $\pi^* = \pi$. In practice the rollouts come from $\pi^*_{\text{old}} \neq \pi_{\text{old}}$ and we deploy π^*_θ, giving the actual objective $J(\pi^*_\theta) = \mathbb{E}_{y\sim\pi^*_\theta}[R(y)]$. Repeating the analysis with $\pi \to \pi^*$, the drift that governs RFPO's bias — and that the trust region should bound — becomes the *deployed drift* $\pi^*_\theta - \pi^*_{\text{old}}$. Computing it during training would require reloading the rollout engine after each actor update, which is expensive, so we estimate it instead. Section 4.3 shows the decoupled anchor $\Delta(y) = \pi_\theta - \pi_{\text{old}}$ is the faithful, cheaper estimate (the actor is already loaded in the training engine to compute π_{old}), whereas the bypass anchor $\Delta^*(y) = \pi_\theta - \pi^*_{\text{old}}$ is cheapest (no recompute) but mixes the two engines and is noisier — why bypass is the less stable trust region. GRPO's bypass-mode ratio $\pi_\theta/\pi^*_{\text{old}}$ carries the same cross-engine mix but, being multiplicative, turns it into exploding variance — the worst collapse (Section 5.2).

4.3 THREE MECHANISMS FOR STABILITY AND QUALITY

The surrogate J_{RF} alone does not guarantee stability — with the wrong choice of $A(y)$ the loop still collapses (the naive RS-SFT case is the simplest example). Three mechanisms, each motivated by the analysis above, make J_{RF} both well-defined and faithful to J.

M1: z-score advantage reweighting. For prompt x_i and rollout $y_{i,j}$, set $A_{i,j} = \frac{R(x_i,y_{i,j})-\mu_i}{\sigma_i}$, where μ_i, σ_i are the mean and standard deviation of rewards within the prompt's group of N rollouts; prompts with $\sigma_i = 0$ are skipped. The per-prompt total of the positive $A_{i,j}$ is bounded by $N/2$ (as is the absolute total of the negatives). The mass peaks at uncertain prompts ($k = N/2$ correct out of

N). M1 thus *balances the contribution from each prompt*: no easy (or hard) prompt can dominate the update, and genuinely uncertain prompts — where the learning signal is richest — contribute the most. This is exactly GRPO's group-relative advantage estimator. In RFPO we additionally scale $A_{i,j}$ by $2/N$, so that the bound on the per-prompt total positive (and total negative) advantage is 1, decoupling it from N. This is a constant rescaling of the gradient and does not change training under a scale-invariant optimizer such as AdamW; we verify this empirically in Section G.

M2: negative samples (mean-zero advantage). Because $A_{i,j}$ is centered per group, $\sum_j A_{i,j} = 0$ and positive-advantage samples are balanced by negative-advantage ones, whose MLE term *maximizes* NLL, pushing the policy away from below-mean outputs. This is required by the surrogate itself: with $A^+ = \max(A, 0)$ the objective reduces to $J^+_{\text{RF}} = \mathbb{E}_{\pi^*_{\text{old}}}[A^+ \log \pi_\theta]$, whose optimum on a batch re-weights the observed positives, $\pi_\theta(y_j) \propto A^+_j$, with no mass elsewhere; refitting round after round on samples of the previous fit can only shrink this support, drifting toward a point mass — the formal counterpart of the iterative RS-SFT collapse. M2 is what makes J_{RF} a *well-defined* objective.

M3: absolute per-token trust region. Because J_{RF}'s gradient is a sum of per-token score-function gradients (Section 4.1), we can impose the trust region independently on each token. We mask the gradient on token t when the per-token policy drift $\Delta(y_t) = \pi_\theta(y_t) - \pi_{\text{old}}(y_t)$ — the token-level counterpart of the bias deviation in (2) — exceeds δ in the advantage-improving direction:

$$\rho_{i,j,t} = 0 \;\text{ if }\; \operatorname{sign}(A_{i,j})\,\Delta(y_t) > \delta, \;\text{ and 1 otherwise.} \tag{3}$$

Positives clip the upper bound and negatives the lower; a negative token with $\pi_{\text{old}}(y_t) \le \delta$ cannot fall by more than δ anyway, so the mask is needed only on already-likely tokens — elsewhere the probability floor keeps the drop within δ. Capping Δ locally controls the policy-drift bias of (2); this is the *decoupled* (default) anchor, and π_{old} is cheap — the actor is already loaded, so we only recompute the snapshot. Absolute clipping is scale-independent on the per-token probability (rare tokens get the full upward δ budget, unlike ratio clipping whose budget shrinks linearly in p^{old}; Section C compares both empirically and analytically).

GRPO uses the same z-score advantage (M1), trains on all samples including $A < 0$ ones (M2), and bounds the importance ratio via ratio clipping (M3). Its reference model and KL penalty are auxiliary — useful for anchoring to initialization but not part of its core mechanism; removing the KL term does not prevent GRPO's collapse under drift (Section 5.1). The difference between RFPO and GRPO is not *which* mechanisms they use but *how* they realize them: a weighted-MLE update with absolute per-token clipping in RFPO, an importance-corrected policy gradient with ratio clipping in GRPO. In the strict on-policy setup ($\texttt{mbs} = \texttt{bs}$), we have $r_t \equiv 1$, GRPO's clipped surrogate reduces to A on every token, and RFPO's clip mask is identically inactive. With the same token-mean loss aggregation (which both our implementations use) and no KL term, the per-token gradients are then identical up to M1's constant $2/N$ factor (Section G); our GRPO runs keep verl's $\beta = 0.001$, the only other difference at $K = 1$. The two strategies agree in this trivial regime and diverge as π_θ drifts off π_{old} in the practical regime (Section 4.2).

4.4 The RFPO algorithm

RFPO plugs into Algorithm 1 by specifying UPDATE. Algorithm 2 defines it: M1–M3 compose into a single mini-batch update routine. The advantages in step 1 are computed once per training batch (over the full set of N rollouts per prompt) and reused across mini-batches; the clip mask in step 2 is recomputed per mini-batch since π_θ has drifted off π_{old} since the snapshot.

Common PPO/GRPO components absent in RFPO: no reference model, no importance ratio $\pi_\theta/\pi_{\text{old}}$, no KL term, no value function (PPO only). π_{old} enters only through the clip mask — a binary $\{0, 1\}$ token decision — so no gradient flows through it. Algorithm 1 with this UPDATE defines the full RFPO training procedure.

5 Experiments

We train Qwen3-4B (instruct) (Qwen Team, 2025) on POLARIS-53K for math, and on DeepCoder verified 16K (PrimeIntellect's SYNTHETIC-1) for coding, $\texttt{bs}$=256 prompts, $N = 16$ rollouts at

Algorithm 2 UPDATE$_{\text{RFPO}}(\pi_\theta, \pi_{\text{old}}, m)$ — the RFPO update routine for Algorithm 1

Require: Policy π_θ, snapshot π_{old}, mini-batch m, clip threshold δ, learning rate η

1: **Advantages (M1, M2).** For each prompt i in m: skip if $\sigma_i = 0$; else $A_{i,j} \leftarrow \frac{2}{N} \frac{R(x_i, y_{i,j}) - \mu_i}{\sigma_i}$.
2: **Clip mask (M3).** For each token, set $\rho_{i,j,t} = 0$ if $\text{sign}(A_{i,j})\, \Delta(y_{i,j,t}) > \delta$, else $\rho_{i,j,t} = 1$; bypass mode replaces Δ with Δ^*.
3: **Weighted-MLE loss.** $\mathcal{L} \leftarrow -\sum_{i,j,t} \rho_{i,j,t}\, A_{i,j} \log \pi_\theta(y_{i,j,t})$, normalized by $\sum \rho_{i,j,t}$.
4: **Gradient step.** $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$.
5: **return** π_θ.

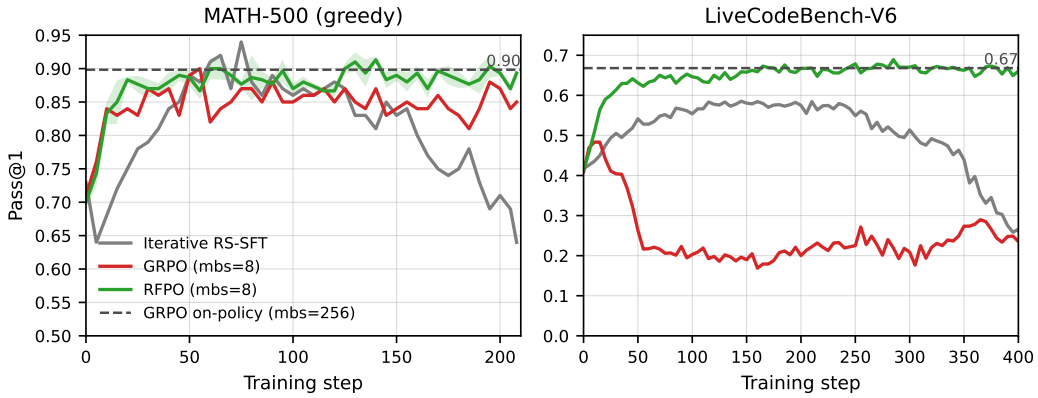


Figure 1: Within-batch drift at mbs=8 ($K = 32$). Validation MATH-500 and LCBv6 (greedy) over training. RFPO (green) holds the strict-on-policy target (dashed line, GRPO mbs=256), while GRPO (red) degrades on math and *collapses* on coding, and the positive-only iterative RS-SFT baseline (gray) peaks then declines. The RFPO math curve shows the 3-seed band.

temperature 1.0, max response length 4096/8192 for math/coding, lr $= 10^{-6}$, binary reward (1 if the boxed answer matches ground truth for math, or the code passes the unit tests for coding); see Section A for the full training configuration and the verl patch. We sweep mbs from 256 (mbs=bs, $K = 1$, strict on-policy) down to 8 ($K = 32$, heavy policy drift). We evaluate the math-trained model on a fixed, uniformly random 100-problem subset of MATH-500 (Hendrycks et al., 2021), for fast training-time validation (denoted MATH-500 below), and the coding trained model on LiveCodeBench-V6 (LCBv6). Validation uses greedy decoding.

5.1 STABILITY TEST 1: POLICY DRIFT (MBS SWEEP)

At strict on-policy (mbs=bs$= 256$, $K = 1$) the two updates are equivalent by construction (Section 4.3), and GRPO serves as the strict-on-policy reference (≈0.90/0.67 on MATH-500/LCBv6), sustaining through training without the collapse of the naive RS-SFT baseline. The question is whether RFPO retains this as drift grows. We sweep mbs from 256 ($K = 1$) down to 8 ($K = 32$): smaller mbs $=$ larger K $=$ more policy drift $=$ larger $\mathbb{E}[r_t^2]$ for GRPO (Section 4.2), while RFPO's bounded-bias estimator is unaffected.

The pattern in Table 1 matches Section 4.2's prediction. At low drift the methods match, and across the three math seeds this is robust rather than a one-off: RFPO reaches 0.91 ± 0.02 (mbs=8) and 0.898 ± 0.004 (mbs=32) against strict-on-policy GRPO's 0.90 ± 0.01, with overlapping seed intervals. At matched mid-drift mbs=32, GRPO degrades by ∼3 points in math and ∼28 points in coding, while RFPO holds at both mbs; this cleanly isolates the IS-vs-surrogate trade-off because both methods see the same policy drift. At heavy drift mbs=8, GRPO degrades a further ∼2 points in math and ∼15 more in coding (to 0.24), while RFPO stays at the on-policy level (Figure 1). Figure 2 directly links GRPO's performance drop to its rising policy drift $\pi_\theta - \pi_{\text{old}}$ as mbs shrinks. The KL term is not what drives it: an otherwise identical coding run at mbs=8 with $\beta = 0$ collapses to the same level (0.214 vs. 0.206 , against RFPO's 0.644).

Table 1: Policy drift (the `mbs` sweep) and rollout drift (bypass), MATH-500 and LCBv6. *Default* columns sweep `mbs` (down the table = more policy drift); *bypass* columns add rollout drift at matched `mbs`. GRPO degrades down the sweep and *collapses* under bypass (its bypass cells fall far below the strict-on-policy target), while RFPO stays flat along the sweep and is far more robust under bypass. Values are late-means taken where strict on-policy GRPO converges (MATH-500 over steps 185–208, ∼1 epoch, 3 seeds; LCBv6 over 240–270, ∼4.5 epochs, single seed); ± is the 3-seed standard deviation, and cells without it are single seed. GRPO's `mbs`=8 LCBv6 bypass cell is mid-collapse in this window (Section 5.2).

			default		bypass	
Method	`mbs`	K	MATH-500	LCBv6	MATH-500	LCBv6
GRPO	256	1	0.90 ±0.01	0.667	—	—
GRPO	32	8	0.866 ±0.013	0.384	0.74	0.041
GRPO	8	32	0.848	0.235	0.75	0.294
RFPO	32	8	0.898 ±0.004	**0.672**	**0.895**	**0.669**
RFPO	8	32	**0.91** ±0.02	0.669	0.87	0.568

Table 2: Mechanism ablation on math training (`mbs`=8; MATH-500 late-mean). *Top block* builds RFPO up from positive-only iterative RS-SFT one mechanism at a time; *bottom block* removes a single mechanism from the full recipe. M1 alone only delays collapse ($0.71 \rightarrow 0.78$); adding M2 without M3 collapses (0.61, grad-norm CoV explodes to 4.97); the full recipe is stable (0.91). Removing M2 or M3 from the full recipe collapses (0.59/0.61) while removing M1 costs only ∼2 points but creates spikes in grad-norm. `val_rep` = mean 20-char repetition rate; `grad CoV` = coefficient of variation of `actor/grad_norm` over training.

Configuration	M1	M2	M3	MATH-500	val_rep	grad CoV
RS-SFT (positive-only)	—	—	—	0.71	0.55	0.34
+ z-score (M1)	✓	—	—	0.78	0.30	0.29
+ negatives (M1+M2)	✓	✓	—	0.61	0.49	4.97
+ clipping = RFPO	✓	✓	✓	0.91	0.08	0.18
RFPO − negatives (M2)	✓	—	✓	0.59	0.57	0.33
RFPO − z-score (M1)	—	✓	✓	0.89	0.06	0.46

5.2 Stability test 2: rollout drift (bypass mode)

The default recipe recomputes π_{old} with the training engine, so the actor and its clip anchor share numerics — the faithful setting of Section 4.2. Verl's *bypass* mode instead reuses vLLM's rollout log-probabilities as π_{old}: faster, since it skips a forward pass over every rollout, but no longer the correct trust-region anchor — it substitutes π^*_{old} for π_{old}, mixing the two engines. The *bypass* columns of Table 1 report both methods at matched `mbs`.

Both pay for the mixed engine, but very differently. RFPO degrades mildly — unchanged at `mbs`=32 ($0.898 \rightarrow 0.895$ MATH-500, $0.672 \rightarrow 0.669$ LCBv6) and 4–10 points at `mbs`=8 ($0.91 \rightarrow 0.87$; $0.669 \rightarrow 0.568$) — while GRPO collapses, by ∼12 points at `mbs`=32 ($0.866 \rightarrow 0.74$) and to near zero on LCBv6 (Figure 3); both GRPO coding runs are below 0.05 at step 400, `mbs`=8 collapsing later (crossing 0.1 near step 360 vs. 215), so their order in Table 1 reflects timing, not outcome. In bypass mode GRPO's ratio $r_t = \pi_\theta / \pi^*_{\text{old}}$ divides by the *actual* rollout distribution, more faithful than default mode. But GRPO's multiplicative ratio turns the cross-engine mismatch into variance, whereas RFPO's ratio-free gradient confines it to the binary clip mask. Default is thus the mode to prefer, but RFPO can also take the bypass speedup when throughput matters — at a cost that grows as `mbs` shrinks — an option GRPO does not have.

5.3 Mechanism ablations

Having established that the full RFPO recipe matches or exceeds GRPO and stays stable under drift, we test what each mechanism contributes. Table 2 does this from both directions — building RFPO

up from positive-only iterative RS-SFT ("RS-SFT" hereafter) one mechanism at a time, and removing a single mechanism from the full recipe.

M1 (z-score) smooths optimization dynamics. Adding M1 to the positive-only baseline lowers grad-norm CoV ($0.34 \rightarrow 0.29$) and lifts MATH-500 ($0.71 \rightarrow 0.78$), while removing it from the full recipe raises CoV $2.5\times$ ($0.18 \rightarrow 0.46$, peaking at $5\times$). Its contribution to *stability*, though, is smaller than M2's and M3's — the $+$M1 run still collapses from its peak (Table 2 top block; Figure 4) — so M1 is mainly a quality and smoothness mechanism, the surrogate counterpart of GRPO's group-relative advantage.

M2 (negatives) and M3 (clipping) together provide stability. M2 adds the sign-aware corrective gradient that positive-only training lacks, but on its own — added with M1 but no M3 — it *worsens* collapse (to 0.61) and makes grad-norm variance explode (CoV 4.97): negatives without a trust region cause unbounded-magnitude oscillation. M3 bounds those updates, giving the stable full recipe (0.91). The leave-one-out rows confirm both are necessary: removing M2 reproduces the unidirectional collapse of naive RS-SFT (Section 4.1, 0.59) — silent in grad-norm (CoV 0.33), a drift toward a point mass — while removing M3 reopens the same exploding-variance collapse. Grad-norm smoothness and task stability are therefore distinct axes.

Failed-stabilizer audit. Figure 5 asks the converse: can some *other* regularizer stand in for M2? We add six standard ones — entropy bonuses, a KL penalty to a reference model, EMA-rollout smoothing, surface-level repetition penalties, and ratio/absolute clipping — on top of the positive-only RS-SFT baseline (i.e. J_{RF} with $A = \mathbf{1}[R = 1]$, no M2 or M3). Each reshapes the update's magnitude or spread, but none supplies the sign-aware *corrective* gradient on wrong answers that M2 provides, and none prevents collapse — M2's role cannot be filled by a side regularizer.

5.4 Efficiency

RFPO skips the reference forward, the importance-ratio computation, and the KL loss. At each method's natural recipe this is $\sim 1.3\times$ faster on 8×H100 for math training: $\sim$387 s/step (RFPO mbs=8 default) vs. $\sim$501 s/step (GRPO mbs=256 default). One step is one outer batch (256 prompts $\times$ 16 rollouts), including all inner optimizer steps. At matched `mbs`=8 on coding, RFPO is $\sim 1.86\times$ faster ($\sim$800 s/step vs $\sim$1490 s/step). RFPO can additionally take the bypass speedup ($\sim$9%) at an accuracy cost that grows as `mbs` shrinks, whereas GRPO bypass collapses. GRPO's times include reloading its offloaded reference model (Section A), part of the cost of keeping one.

6 Conclusion and Discussion

We framed on-policy fine-tuning through a bias-variance lens: GRPO importance-corrects and pays variance that grows with drift, while RFPO retargets to the ratio-free surrogate J_{RF} and pays a trust-region-controlled bias. The two coincide at strict on-policy and diverge as drift grows; the prediction holds on both drift axes, GRPO degrading at smaller `mbs` and collapsing under bypass while RFPO stays stable under policy drift and is far more robust under bypass, with its three mechanisms playing distinct variance, well-definedness, and bias-bound roles.

Implications. Surrogates that drop the IS correction can dominate IS-corrected methods in MSE under significant drift, and RL-style training without a reference model, KL penalty, or importance ratio is feasible at the small `mbs` that enables many optimizer steps per batch — and, since RFPO changes only the update step, it is a drop-in replacement for GRPO's update in existing pipelines. Under drift, a controlled bias beats a growing variance.

Limitations and scope. Our study covers Qwen3-4B on two verifiable-reward domains (math, POLARIS-53K; coding, DeepCoder-16K) plus Phi-4-mini-reasoning (Section F); other scales and task families remain untested. We do not compare head-to-head with GSPO, MinPRO or DAPO.

Next steps. First, combining the ratio-free surrogate with a sequence- or prefix-level trust region in the style of GSPO and MinPRO (Section D), since the two are complementary. Second, *long-horizon agentic* training, where both drifts compound across turns.

REPRODUCIBILITY STATEMENT

All experiments use the public Qwen3-4B model and the POLARIS-53K (math) and DeepCoder-16K (coding) datasets (Section 5). RFPO's full update rule is given in Algorithm 2 (Section 4.4), and its three mechanisms M1–M3 — z-scored advantages, negative samples, and an absolute per-token trust region ($\delta = 0.1$) — are defined in Section 4.3. Complete training hyperparameters for both RFPO and GRPO, together with the verl offload-symmetry patch required to reproduce the small-`mbs` GRPO runs, are in Section A (Table 3); the bias-variance derivation is in Section B. Evaluation uses the MATH-500, LiveCodeBench-V6, and AIME 2025 under greedy decoding. We will release our training code and configurations.

AI USE STATEMENT

In this work, we used generative AI tools for two purposes, both of which fall under the *recommended* disclosure category of the ICLR 2027 AI policy. First, we used LLM-based assistants to aid and polish writing: editing prose for clarity and concision, on text whose content, claims, and organization were authored by us. Second, we used LLM-based tools for retrieval and discovery, in particular for searching for related work and surfacing candidate references; every reference so surfaced was located in its original form, read by the authors, and verified before citation.

We did not use generative AI tools for tasks in the policy's *required* disclosure category. The research question and the OPFT loop framing, the RFPO objective and its three mechanisms, the bias-variance analysis and its derivation (Section B), the experimental design, the implementation of RFPO and the verl offload-symmetry patch, and the interpretation of all reported results are the authors' own work.

We have reviewed all AI-assisted work: AI-edited prose was checked line by line against the authors' intended meaning, and every AI-suggested reference was independently retrieved and read before being cited. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Shiye Lei, Zhihao Cheng, and Dacheng Tao. A step back: Prefix importance ratio stabilizes policy optimization. *arXiv preprint arXiv:2601.22718*, 2026.

Yingru Li, Jiacai Liu, Jiawei Xu, Yuxuan Tong, Ziniu Li, Qian Liu, and Baoxiang Wang. Trust region masking for long-horizon LLM reinforcement learning. *arXiv preprint arXiv:2512.23075*, 2025.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pp. 745–750, 2007.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2023.

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

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

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, 2025.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. *arXiv preprint arXiv:2602.02710*, 2026.

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, Shuohang Wang, Weijian Xu, Jianfeng Gao, and Weizhu Chen. Phi-4-Mini-Reasoning: Exploring the limits of small reasoning language models in math. *arXiv preprint arXiv:2504.21233*, 2025.

Qiying Yu et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

A Experiment details

Table 3 lists the shared training configuration; the two methods are identical except the rows noted. Rollouts are generated by vLLM and training runs under FSDP, and validation uses greedy decoding late-meaned over the windows in Table 1 (math cells are 3-seed means; coding and bypass cells are single seed).

verl patch. Running GRPO at small `mbs` under FSDP with parameter offload deadlocked in the reference-model forward: verl offloads the reference module but — unlike the actor — never loads it back before each forward, so with many optimizer steps per batch one rank eventually reads a freed shard and stalls NCCL. A $\sim$10-line fix mirroring the actor's load/offload for the reference module (the "patched" verl) removes the deadlock at all `mbs`; all GRPO runs use it, so the reported regressions reflect algorithmic instability (IS variance under drift), not this operational one.

Table 3: Training hyperparameters. Spanned cells are shared by both methods; only the three rows below the second rule differ. The Phi-4-mini-reasoning runs of Section F are coding-only and train for 406 steps (≈7 epochs); every other entry is as listed.

	RFPO	GRPO
Base model	Qwen3-4B (instruct); Phi-4-mini-reasoning (Section F)	
Training data	POLARIS-53K (math)/DeepCoder-16K (coding)	
Steps	208 (math, 1 epoch); coding ∼4.5 epochs	
Train batch	256 prompts	
Rollouts N	16 @ T=1.0	
Max resp. len.	4096 (math)/8192 (coding)	
LR	10^{-6}	
Negatives	weight 1.0, symmetric (no downweighting)	
Loss aggregation	token-mean over the mini-batch (verl `loss_agg_mode`)	
Clip mode	absolute, $\delta = 0.1$	ratio, $\epsilon = 0.2$
KL penalty	—	$\beta = 0.001$ ($\beta = 0$ control: Section 5.1)
Reference model	none	frozen copy of init

B BIAS-VARIANCE DERIVATION

Throughout this appendix we adopt vanilla GRPO's implicit identity $\pi^*_{\text{old}} = \pi_{\text{old}}$ (Section 4.2), so the sampler coincides with π_{old}, and we idealize $\hat{g}_{\text{IS}}$ as unbiased for ∇J (hence the $\|\nabla J\|^2$ term below), ignoring the residual prefix and continuation mismatch of per-token ratios noted in Section 4.1; this idealization favors GRPO. A rollout drift $\pi^*_{\text{old}} \neq \pi_{\text{old}}$ adds bias to $\hat{g}_{\text{IS}}$ (Section 5.2) but leaves $\hat{g}_{\text{RF}}$ unchanged.

Variance of $\hat{g}_{\text{IS}}$. With $\hat{g}_{\text{IS}} = r_t \cdot A \cdot \nabla \log \pi_\theta$ and $r_t = \pi_\theta / \pi_{\text{old}}$,

$$\text{Var}_{\pi_{\text{old}}}(\hat{g}_{\text{IS}}) = \mathbb{E}_{\pi_{\text{old}}}[r_t^2 \cdot \|A \nabla \log \pi_\theta\|^2] - \|\nabla J\|^2.$$

With $r_t \in [1-\epsilon, 1+\epsilon]$ enforced by the clip, the first term is bounded on a single step by $(1+\epsilon)^2 \cdot \sigma^2_{\text{on-policy}}$. Across K steps within a batch, π_θ moves while π_{old} is held fixed, and the per-token amplification compounds.

Bias of $\hat{g}_{\text{RF}}$. The estimator $\hat{g}_{\text{RF}} = A \cdot \nabla \log \pi_\theta$ is unbiased for ∇J_{RF} but biased for ∇J. Writing $\Delta^*(y) = \pi_\theta(y) - \pi^*_{\text{old}}(y)$ (which reduces to $\pi_\theta - \pi_{\text{old}}$ under this appendix's $\pi^*_{\text{old}} = \pi_{\text{old}}$),

$$\nabla J - \nabla J_{\text{RF}} = \textstyle\sum_y \Delta^*(y) \cdot A(y) \cdot \nabla \log \pi_\theta(y),$$

so $\|\text{Bias}(\hat{g}_{\text{RF}})\| \leq C \cdot \|\Delta^*\|_1 \leq 2C \cdot \text{TV}(\pi_\theta, \pi_{\text{old}})$ with $C = \sup \|A \cdot \nabla \log \pi_\theta\|$. Absolute clipping caps the per-token additive drift on contributing tokens to δ; ratio clipping caps the multiplicative drift to ϵ. Neither cap bounds the sequence-level TV by its threshold: even if every position drifted by at most δ, the telescoping bound would give only $\text{TV} \leq T\delta$, vacuous for long responses, so the bound is informative only for small drift, and we read M3 as a local regularizer of this bias rather than a global guarantee.

Clipping bias. Both recipes also incur a *clipping/masking* bias relative to their own *unclipped* estimators. Practical GRPO is not strictly unbiased because ratio clipping suppresses gradient contributions once the policy moves sufficiently far in the advantage-improving direction. Relative to unclipped importance-weighted gradient:

$$\text{Bias}_{\text{GRPO,clip}} = -\mathbb{E}_{\pi_{\text{old}}}[r_t A \cdot \nabla \log \pi_\theta(y) \mathbf{1}\{A\Delta^*(y) > |A| \epsilon p_{\text{old}}\}].$$

For RFPO, the bias introduced by masking relative to the unmasked surrogate is

$$\text{Bias}_{\text{RFPO,clip}} = -\mathbb{E}_{\pi_{\text{old}}}[A \cdot \nabla \log \pi_\theta(y) \mathbf{1}\{A\Delta^*(y) > |A| \delta\}].$$

Two differences follow. First, the trigger: For tokens with $p_{\text{old}} < \delta/\epsilon$, RFPO has strictly larger allowable directional probability budget than GRPO's earlier clipping. With our $\delta = 0.1, \epsilon = 0.2$ this is $\pi_{\text{old}}(y_t) < 0.5$, i.e. essentially every token but the highest-probability ones. Second, the weight:

GRPO's discarded gradient contributions carry the multiplicative importance factor, which itself grows with drift, whereas RFPO's carries none.

Note that the two clip-bias expressions above are each measured against that method's *own* unclipped estimator, so they are not directly comparable as errors in ∇J. Under the idealization above, GRPO's only bias is $\mathrm{Bias}_{\mathrm{GRPO,clip}}$. RFPO trades the r_t^2 variance for an *additional* surrogate-gap term — which is precisely the term M3 controls locally.

Crossover. For sufficient drift, the variance term in $\mathrm{MSE}(\hat{g}_{\mathrm{IS}})$ exceeds the squared bias of $\hat{g}_{\mathrm{RF}}$, so $\hat{g}_{\mathrm{RF}}$ has lower MSE for ∇J. We do not derive a sharp crossover; the qualitative monotonic-in-drift prediction is empirically tested in Section 5.

C Absolute vs. ratio clipping (M3)

Ratio clipping ($\pi_{\mathrm{new}}/\pi_{\mathrm{old}} \in [1-\epsilon, 1+\epsilon]$) is scale-dependent. Specifically, the allowable absolute probability movement of GRPO is $\epsilon\pi_{\mathrm{old}}$, whereas for RFPO it is δ, and the crossover point is $\pi_{\mathrm{old}} < \frac{\delta}{\epsilon}$. With $\delta = 0.1,\ \epsilon = 0.2$ this crossover is at $\pi_{\mathrm{old}} = 0.5$, so absolute clipping is the more permissive rule on essentially every token except the highest-probability ones. At $\pi_{\mathrm{old}} = 0.001$ and $\epsilon = 0.2$, the maximum allowed p_{new} is 0.0012 — $500\times$ tighter than absolute clipping at $\delta = 0.1$. This matters for positives, which often act on rare tokens that should become common; for negatives, a token with $\pi_{\mathrm{old}} \leq \delta$ cannot fall by more than δ, so the mask is needed only on already-likely tokens. Empirically with all other mechanisms held fixed, absolute clipping yields $\sim$4 points higher sustained MATH-500 than ratio clipping.

D Trust-region granularity and concurrent work

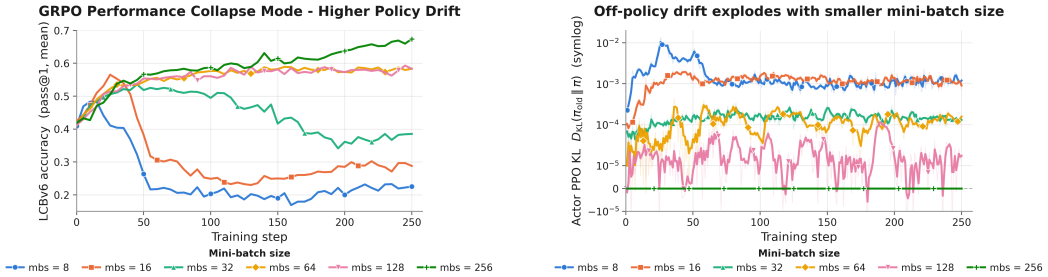
M3 in this paper is a *per-token* trust region: mask the gradient on token t when $\mathrm{sign}(A)\,(\pi_\theta(y_t) - \pi_{\mathrm{old}}(y_t)) > \delta$. Several concurrent works place the trust region at coarser granularities. Zheng et al. (2025) (GSPO) replace token-level importance ratios with a single sequence-level ratio. Lei et al. (2026) (MinPRO) replace the cumulative prefix ratio with the minimum token-level ratio observed in the preceding prefix. Li et al. (2025) bound the maximum token-level divergence across a sequence and mask entire violating sequences. Per-token clipping gives every position a separate budget and admits asymmetric clipping (positives clip up, negatives clip down); sequence- and prefix-level alternatives are a single budget over the whole rollout and may behave differently when either (a) a few tokens dominate the deviation or (b) the deviation is distributed evenly across many tokens. All these placements serve M3's theoretical role of controlling the bias of J_{RF} relative to J (Section 4.2), but they correspond to different points on the bias-variance curve and should be compared empirically. An open question is whether the bias-variance MSE prediction of Section 4.2 sharpens or interacts with Li et al. (2025)'s monotonic-improvement bound when the two are evaluated on the same recipe.

E Additional result figures

This appendix collects supplementary figures for Section 5. Figure 2 expands the coding `mbs` sweep and Figure 3 the bypass-mode collapse (both referenced from the main text); Figures 4 and 5 visualize the mechanism ablations, and Figure 6 reports AIME 2025 (deferred from the main text, which uses MATH-500).

F A second model family: Phi-4-mini-reasoning

All experiments in the main text use Qwen3-4B. To check that the stability gap is a property of the update rule and not of one model family, we repeat the coding configuration on Phi-4-mini-reasoning (Xu et al., 2025) — a different pretraining corpus, tokenizer, and architecture — holding everything else fixed: DeepCoder verified 16K, `bs`$= 256$ prompts, $N = 16$ rollouts at temperature 1.0, max response length 8192, lr $= 10^{-6}$, binary unit-test reward, 406 training steps ($\approx$7 epochs), single seed (42), evaluated on LCBv6 with greedy decoding.



(a) LCBv6 accuracy

(b) KL divergence between π_θ and π_{old}

Figure 2: Coding GRPO across mini-batch sizes. As `mbs` decreases (more optimizer steps per batch, i.e. more policy drift), LCBv6 accuracy falls (a) in step with a rising π_θ–π_{old} KL (b) — the drift-to-variance mechanism of Section 4.2 measured directly.

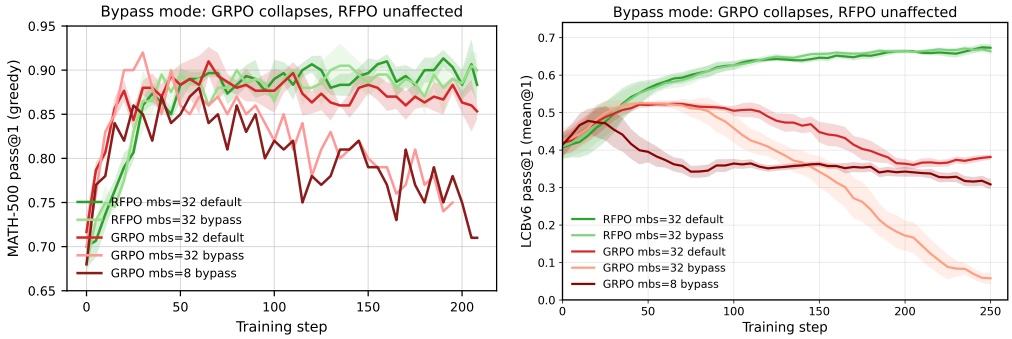


Figure 3: Bypass-mode training curves on MATH-500 and LiveCodeBench v6 (greedy pass@1; RFPO shown at `mbs`=32). On both benchmarks RFPO bypass stays close to RFPO default, while both GRPO bypass runs (`mbs`=32 and `mbs`=8) peak near step 30 then decline monotonically as GRPO default sustains. On MATH-500 the bypass runs fall to 0.71–0.75; on LiveCodeBench v6 the collapse is far more severe—over steps 240–270 (the Table 1 window) GRPO bypass averages 0.04 at `mbs`=32 and 0.29 at `mbs`=8, which collapses later, versus RFPO's stable ≈0.66. The collapse comes from the actor-rollout mismatch entering GRPO's *multiplicative* importance ratio; RFPO's ratio-free gradient admits it only through a bounded clip mask. At step 400, GRPO in bypass mode collapses further to below 0.05 for both `mbs`=32 and `mbs`=8.

Figure 7 and Table 4 show the same ordering as Section 5.1. RFPO is insensitive to `mbs`: at `mbs`$= 8$ ($K = 32$) and `mbs`$= 32$ ($K = 8$) it rises smoothly from 0.23 to a late-mean of 0.494 and 0.499 — the two curves are within seed-scale noise of each other despite a $4\times$ difference in drift. GRPO is not. At `mbs`$= 32$ it climbs slowly to 0.314, and at `mbs`$= 8$ it peaks at 0.285 by step 25 and then *loses* that gain, ending at 0.229 — no better than the untrained model. The drift diagnostics identify the mechanism: at `mbs`$= 8$ GRPO's π_θ–π_{old} KL spikes to 0.134 (nearly $100\times$ its own median of 1.7×10^{-3}) around steps 88–92 and 125–126, exactly where the reward curve turns over, and its token entropy swings over $[0.024, 1.09]$; at `mbs`$= 32$ the same KL stays below 1.4×10^{-3} throughout and entropy decays smoothly within $[0.19, 0.43]$.

The positive-only iterative RS-SFT baseline is unstable on this model too, and in the characteristic way: it improves for 75 steps (to 0.283), then collapses to 0.034. Its own diagnostics show the degenerate fixed point of Section 4.3 directly — mean response length falls from 4892 to 978 tokens while the NLL of its *own* selected samples *rises* from 0.42 to 0.63, i.e. the policy is concentrating on progressively shorter outputs that it fits progressively worse. RFPO's mean response length stays flat (≈5000 tokens) over the same window.

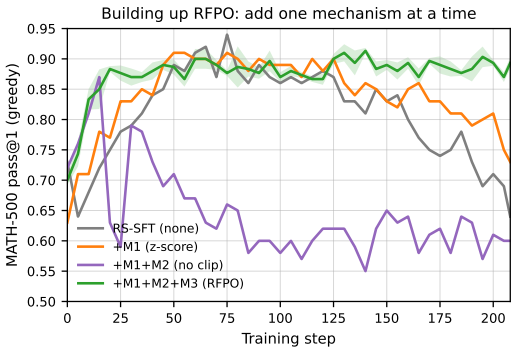


Figure 4: Cumulative ablation (MATH-500, greedy): building RFPO from iterative RS-SFT by adding one mechanism at a time. M1 alone delays collapse (late-mean $0.71 \rightarrow 0.78$) but does not prevent it; M2 without M3 collapses harder (to 0.61 — negatives need a trust region); only the full M1+M2+M3 recipe stays stable (0.91, 3-seed band). Partial configs are single seed.

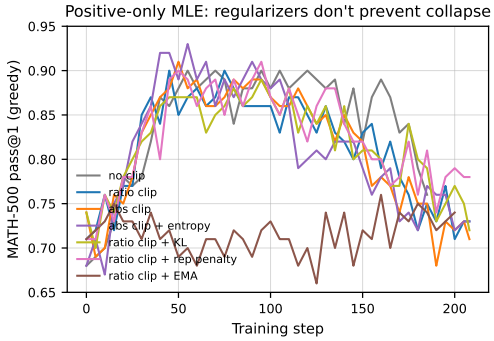


Figure 5: Failed-stabilizer audit (MATH-500, single seed): the positive-only iterative RS-SFT baseline plus a range of regularizers, all of which peak then decline — none recovers stability. Final MATH-500 — no clip 0.76, ratio clip (ϵ=0.2) 0.74, absolute clip (δ=0.1) 0.72, +entropy 0.74, +KL-to-ref 0.75, +rep. penalty 0.78, +EMA rollout 0.73 — all far below the full RFPO recipe's ≈ 0.91, confirming that M2's corrective gradient cannot be substituted by a side regularizer.

G Loss scaling does not affect training

RFPO's update sums a per-token loss over the N rollouts of each prompt group, a variable mix of positives and negatives. Whether the advantages carry M1's $2/N$ factor — equivalently, whether the per-group loss is normalized by $N/2$ — is an implementation choice: it rescales the gradient by a constant factor that the optimizer's adaptive step size largely absorbs. To confirm that our results do not depend on this choice, we ran the coding configuration (DeepCoder-16K, Qwen3-4B, `mbs`$= 8$) twice, identical in every logged hyperparameter except the scaling flag.

Figure 8 shows the two learning curves over the first 250 steps. They are indistinguishable: both rise from 0.41 at initialization to ≈ 0.66, with a mean absolute difference of 0.013 across the 51 logged evaluation points — the same order as the seed-to-seed variation reported in Table 1. Over the final ten evaluations the two settings average 0.664 (with scaling) and 0.655 (without). The rescaling is not a no-op at the gradient level — the median logged gradient norm differs by $7.6\times$ between the two runs (2.2×10^{-3} with scaling, 1.7×10^{-2} without) — so this is an empirical check on AdamW's scale invariance rather than a restatement of it, which is worth making because ϵ, decoupled weight decay, and global gradient-norm clipping all break that invariance exactly. We therefore report results without distinguishing the two, and the choice is not one of RFPO's three mechanisms.

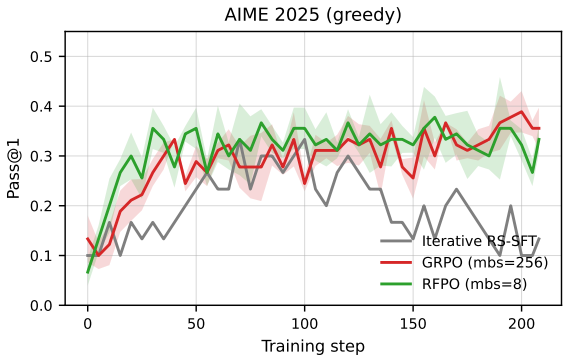


Figure 6: AIME 2025 (greedy), math training. RFPO (mbs=8, 3-seed band) matches GRPO (mbs=256, 3-seed band) while the positive-only iterative RS-SFT baseline collapses. GRPO is shown only at strict on-policy. Deferred from the main text, which reports MATH-500.

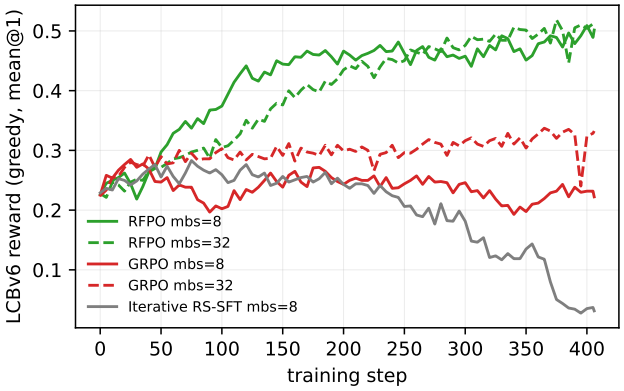


Figure 7: Phi-4-mini-reasoning on DeepCoder-16K, LCBv6 validation (greedy). RFPO (green) is flat in mbs: the 8 and 32 curves coincide. GRPO (red) is not: mbs$=32$ underperforms RFPO by $\approx$18 points and mbs$=8$ decays back to its initialization. The positive-only iterative RS-SFT baseline (gray) peaks at step 75 and then collapses.

Table 4: Phi-4-mini-reasoning, LCBv6 (greedy, mean@1), single seed. *Peak* is the best evaluation and *late* the mean over the final 25 steps; a large peak–late gap is the signature of instability. RFPO has none at either mbs; GRPO at mbs$=8$ gives back its entire gain, and iterative RS-SFT collapses.

Method	mbs (K)	init	peak (step)	late
RFPO	32 (K=8)	0.228	0.519 (375)	**0.499**
RFPO	8 (K=32)	0.226	0.509 (400)	**0.494**
GRPO	32 (K=8)	0.224	0.337 (365)	0.314
GRPO	8 (K=32)	0.225	0.285 (25)	0.229
Iterative RS-SFT	8 (K=32)	0.229	0.283 (75)	0.034

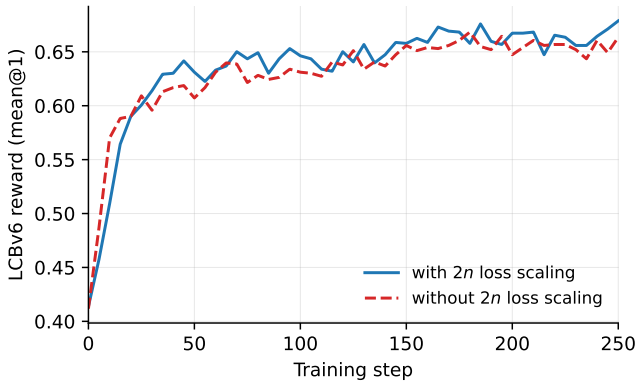


Figure 8: Loss scaling does not affect training (LCBv6, DeepCoder-16K, Qwen3-4B, $\texttt{mbs}= 8$, first 250 steps). The two runs are identical in every logged hyperparameter except whether the advantages carry the $2/N$ factor of M1; their curves coincide to within seed noise (mean absolute difference 0.013 over 51 evaluation points).