UNDERSTANDING ENRICHMENT IN REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

When rewards are sparse, reinforcement learning with verifiable rewards (RLVR) often uses hints or intermediate guidance to generate more successful rollouts. This enrichment biases policy-gradient updates unless corrected via importance weights, but existing methods omit correction or truncate importance weights in order to avoid the high variance of correction. It thus remains unclear what exactly is gained or lost in RLVR by correcting enriched rollouts. We show, mathematically, that omitted or truncated correction implicitly reweights the defined reward, and we decompose the resulting gradient error into scale, rotation, and variance to explain their distinct effects on learning. To make correction practical, we develop a novel sequential Monte Carlo (SMC) weight correction mechanism that, under stability and particle-order assumptions, tempers the exponential compounding of standard correction variance over the length of a sample to an additive accumulation. We then apply our analysis of enrichment to interpreting the results of a fine-tune of Qwen3-1.7B on a sparse band of OpenMathReasoning, establishing a concrete mechanism of how enrichment, both corrected and uncorrected, helps avoid collapse in sparse domains. Our main contribution is to understand, in general, how enrichment and correction can affect training, as opposed to claiming that either mode of operation is superior to unenriched RL.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has produced strong results in a wide variety of domains (Lambert et al., 2024; Guo et al., 2025; Wei et al., 2026; Ren et al., 2025; Jin et al., 2025), its primary advantage being that the model can learn from its own generations and a deterministic verifier. However, RLVR struggles when the policy initially has a low success rate: verifier rewards are often binary (e.g., 1 if a math problem is solved correctly, 0 if not), and thus an incompetent policy often ends up generating many zero-reward samples (or entire groups) offering no learning signal (Zhang et al., 2026a). Failure to generate a learning signal forces more rollouts during training, making RLVR inefficient in some cases (Le et al., 2026; Foster et al., 2025).

A common response to this problem is to *enrich* the policy with external guidance that makes successful trajectories more probable. Several methods supply hints or solution prefixes and generate rollouts under the hinted context (Agrawal et al., 2026; Zhang et al., 2026b; Huang et al., 2025; Zhang et al., 2025; Li et al., 2026a). Others use process feedback to guide tree-structured rollouts: PaTR (Li et al., 2026b), for example, uses process scorers, including pretrained process reward models (PRMs), to expand promising partial trajectories and stop unpromising branches early, then trains on the resulting rollout groups with GRPO using task-outcome rewards.

A key observation is that such enriched policies *alter the distribution* from which samples are drawn. Calling the non-enriched base policy π_θ for a set of parameters θ (e.g., the parameters of an LLM), the goal of reinforcement learning (RL) is to maximize the expectation of a reward function R on samples x drawn from π_θ, whose gradient is:

$$g = \nabla_\theta J(\theta) = \mathbb{E}_{x\sim\pi_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)] \tag{1}$$

Computing a gradient u over samples from an enriched policy q_θ instead of π_θ changes its value:

$$u = \mathbb{E}_{x\sim q_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)] \tag{2}$$

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.

Clearly $u \neq g$ in general, so updating θ with the enriched gradient u can lead to a different training outcome. It is possible to *correct* u, by weighing the contribution of each sample to counterbalance the difference in sampling rate: computing $\mathbb{E}_{x\sim q_\theta}[\frac{\pi_\theta(x)}{q_\theta(x)} \cdot R(x)\nabla_\theta \log \pi_\theta(x)]$ instead of Equation (2).

However, many existing enrichment-based RLVR methods (Li et al., 2026a; Zhang et al., 2026b; 2025) do not compute the correction, or compute only a clipped approximation (Agrawal et al., 2026). Two reasons help explain this practice. First, full correction obtained by multiplying the correction weight $\pi_\theta(x)/q_\theta(x)$ often results in significant variance when using a finite number of samples to estimate g during training, especially for long-horizon samples generated using LLMs (Shen et al., 2026a). Second, perhaps more importantly, existing work has shown that uncorrected or partially corrected gradients, where the correction weight $\pi_\theta(x)/q_\theta(x)$ is truncated to stay within a fixed range, still often lead to good results. Consequently, most existing work focuses on developing methods of enrichment whose benefits hold under uncorrected or partially corrected gradients, as opposed to directly investigating the effects of weight correction on training or methods to reduce variance correction. This leads to the central question we aim to answer in this paper:

What does enrichment do to RL training, and what is gained or lost by correcting for it?

Our paper primarily investigates this question through a mathematical analysis that characterizes enrichment as *implicitly modifying the objective* used by RL, and correction as a process that recovers the declared objective (Section 2.1). We also observe that training on the modified objective does not necessarily always lead to a worse policy. The reweighted rewards manifest in training as a bias in the expected gradient, and we decompose the bias into scale and rotation terms, along with an additional variance term induced by finite estimation and correction, and isolate the effect that each element has on training through controlled experiments (Sections 2.2.1 and 2.2.2).

Our theory suggests that full correction may be worth investigating as an alternative for uncorrected enrichment in RL, but we observe that in LLMs-as-policies, the variance of the correction weights can be so high that standard correction results in computations that may not be stably representable in float32 (Shen et al., 2026a; Sheng et al., 2025). We thus develop a method for drawing rollouts from the enriched policy q_θ while keeping the variance cost of correction low, based on sequential Monte-Carlo sampling (SMC) (Doucet et al., 2001). A key feature of our SMC-correction is an *additive-in-length* variance bound (under some mathematical assumptions in Section B.2) in contrast to the exponential bound of standard weight correction, and to the best of our knowledge, is the first approach that enables practical, untruncated weight correction for real RL training.

Finally, we fine-tune Qwen3-1.7B (Yang et al., 2025a) on a sparse-success subset of OpenMath-Reasoning (Moshkov et al., 2025), via (*i*) uncorrected enriched, (*ii*) SMC-corrected enriched, and (*iii*) unenriched RL as an application of our theory to analyze a real RL run (Section 4). Our results identify a concrete mode of training where enrichment, including the uncorrected variant, becomes useful: when the base policy has a behavioral profile that has a small number of successes concentrated on a few problems, unenriched base RL collapses towards them. Enrichment can expose the policy towards a more diverse set of successes and delay the collapse or avoid it altogether.

Overall, our paper is positioned as a study of enrichment and its correction, providing a theoretical basis for understanding enrichment and correction that future work can build upon rather than a novel algorithm that achieves state-of-the-art results.

2 How Enriched Sampling Changes Learning

We now investigate how enrichment impacts learning from a mathematical perspective. Intuitively, uncorrected enrichment, in the ideal case, improves learning by implicitly *amplifying* the reward assigned to samples that occur more frequently in the enriched policy compared to the base policy. Operationally, this difference manifests itself as a bias on the expected gradient, which can further be decomposed into two main elements: (*i*) *scale* on the size of the gradient and (*ii*) *rotation* on the direction, with an additional error term introduced due to (*iii*) *variance* of the estimator.

Setting and notation. In this paper, we use π_θ to denote the base policy (without enrichment) and q_θ to denote the enriched policy, where θ refers to the shared set of parameters that define π_θ and q_θ. We use x to denote a complete response from a policy p_θ, $p_\theta(x)$ to denote the probability of drawing x from p_θ, and $R(x)$ to denote the reward of x. We focus on LLMs-as-policies where a

sample x is a series of tokens $x = t_1, \cdots, t_l$, abbreviated as $t_{1:l}$, and a policy p_θ computes next-token probabilities given a prefix (i.e., $p_\theta(t_i \mid t_{1:i-1})$). We focus in this section on standard REINFORCE learning (Williams, 1992); an extension to GRPO (Shao et al., 2024) can be found in Section A.

2.1 Explaining enrichment as a redistribution of reward

Recall Equations 1 and 2, which compute the precise gradients when the sample is drawn from the base policy π_θ and the enriched policy q_θ:

$$g = \mathbb{E}_{x\sim\pi_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)] \quad (1) \qquad\qquad u = \mathbb{E}_{x\sim q_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)] \quad (2)$$

In practice, g and u are *estimated* via a set of N independent samples $x_1, \cdots, x_N$ drawn from their respective policies. Calling the policy in question p_θ, in both cases the inner term $R(x)\nabla_\theta \log \pi_\theta(x)$ expands to produce the following estimation over the N samples:

$$\hat{h} = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{\nabla_\theta \pi_\theta(x_i)}{\pi_\theta(x_i)} \cdot R(x_i) \qquad\qquad (x_i \sim p_\theta). \tag{3}$$

The mismatch comes from the denominator $\pi_\theta(x_i)$ and the distribution from which x_i is drawn from. If $x_i \sim \pi_\theta$, then $\pi_\theta(x_i)$ cancels out in expectation, and the mean of the unenriched estimator $\hat{g}$ becomes $\sum_x \nabla_\theta \pi_\theta(x) \cdot R(x)$: intuitively, to move the parameters θ in a direction such that the probability ($\pi_\theta(x)$) of drawing a rewarding sample (e.g., $R(x) = 1$ for binary rewards) is increased.

On the other hand, if $x_i \sim q_\theta$, the cancellation does not happen and the intuitive reading of the gradient no longer holds. In expectation, the estimator of the uncorrected gradient $\hat{u}$ instead becomes $\sum_x \frac{\nabla_\theta \pi_\theta(x)}{\pi_\theta(x)} \cdot R(x) \cdot q_\theta(x) = \sum_x \nabla_\theta \pi_\theta(x) \cdot \frac{q_\theta(x)}{\pi_\theta(x)}\ R(x)$: the gradient points in a direction that increases the base policy π_θ's expected *distorted* reward $R'_\theta(x) = \frac{q_\theta(x)}{\pi_\theta(x)} R(x)$, instead of the defined reward $R(x)$. We thus observe that the effect of enrichment can be explained as an implicit *reweighting* of the reward R, where the reward of a sample x is *amplified* if it occurs more frequently in the enriched policy q_θ compared to the base policy π_θ, and *attenuated* in the reverse case.

Example 1 (Unequal reinforcement of successful responses)**.** Consider an example task with two correct responses x_a, x_b and one incorrect response x_f, whose rewards are $1, 1, 0$. Suppose the base policy π_θ assigns them probabilities $(0.01, 0.04, 0.95)$, while the enriched policy q_θ assigns the probabilities $(0.10, 0.02, 0.88)$. The effective rewards are then

$$R'_\theta(x_a) = \frac{0.10}{0.01} \cdot 1 = 10, \qquad R'_\theta(x_b) = \frac{0.02}{0.04} \cdot 1 = \frac{1}{2}, \qquad R'_\theta(x_f) = 0.$$

Enrichment raises the overall reward hit rate from 5% to 12%, but also reweights the effective rewards of x_a and x_b: x_a now has a reward 20 times larger than that of x_b. Compared to the unenriched gradient, enrichment amplifies the contribution that pushes toward x_a tenfold while halving the contribution that pushes toward x_b, despite both responses receiving the same verifier reward.

This reweighting can *help* when enrichment assigns high probability to successful responses that the base policy rarely generates, increasing their training signal; i.e., the reweighting has the effect of teaching the model the successes of the enriched policy as opposed to reinforcing the base policy's own successes (we will see a case of this phenomenon in Section 4). Conversely, enrichment that recognizes only a familiar solution family can disproportionately reinforce the family and steer learning away from alternatives, including higher-reward solutions as illustrated in Section 2.2.2.

2.2 Decomposing Gradient Error into Scale, Rotation, and Variance

We observe that the modified reward R'_θ in Section 2.1 is dependent on the parameters θ, and thus one cannot simply say uncorrected enriched training is equivalent to REINFORCE training under a different fixed reward: the reweighted reward changes each step. Thus the best way to understand the impact of implicit reweighting is through the possible gradient distortions reweighting gives rise to at each update during training: an effect that can be decomposed into scale, rotation, and variance.

Let $g \neq 0$ denote the true gradient defined in Equation (1), u an uncorrected gradient, and $\hat{u}$ an estimator of u such that the mean of the estimator $\mathbb{E}[\hat{u}] = u$. We observe that $\hat{u}$ admits a standard

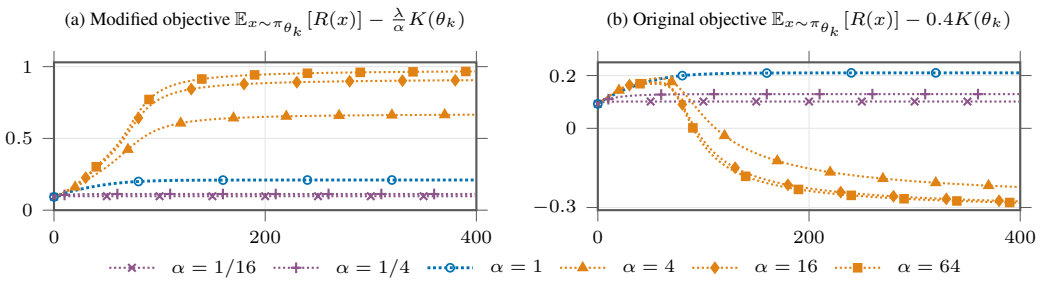


Figure 1: Scale-modified (left, y-axis) and original objectives (right, y-axis) tracked across training steps k (x-axis), for the scale-isolated simulated training runs, over different values of the scale coefficient α. We use θ_k to denote the parameters at step k.

mean-squared-error decomposition against g with bias and variance terms. The bias term can be further split into parallel (a *scale* α on g) and perpendicular (a *rotation* r relative to g) components:

$$\mathbb{E}\|\widehat{u} - g\|^2 = \underbrace{(\alpha-1)^2\|g\|^2}_{\text{scale error}} + \underbrace{\|r\|^2}_{\text{rotation error}} + \underbrace{\mathbb{E}\|\widehat{u} - u\|^2}_{\text{sampling variance}}. \tag{4}$$

Here $\alpha = \langle u, g\rangle / \|g\|^2$ is the signed scale factor and $r = u - \alpha g$ is perpendicular to g. Each term in Equation (4) has a different consequence for training, which we isolate and illustrate the effect of via controlled experiments, below.

2.2.1 Scale: amplifying the learning signal

A *scale* factor arises when the enrichment q_θ consistently prefers samples x where the reward $R(x) > 0$ by a factor of α: i.e., $\forall x, q_\theta(x) = \alpha\pi_\theta(x)$. The derivation in Section 2.1 shows that such consistent oversampling results in an amplified (or attenuated, if $\alpha < 1$) reward $R'_\theta = \alpha R$. If training is unregularized, in expectation, a scale factor does not affect training outside of an increase in how quickly training proceeds and its stability depending on step size.

The interesting difference comes when training is *regularized*: for example, by a KL-penalty term $K(\theta)$, such that the objective $J(\theta)$ becomes $\mathbb{E}_{x\sim\pi_\theta}[R(x)] - \lambda \cdot K(\theta)$ instead of the unregularized expected reward $\mathbb{E}_{x\sim\pi_\theta}[R(x)]$. Uncorrected enrichment amplifies the reward in $J(\theta)$ but not the KL-penalty term, and the mean of the uncorrected estimator $\mathbb{E}[\hat{u}]$ becomes in expectation:

$$\mathbb{E}[\hat{u}] = \alpha g - \lambda\nabla_\theta K(\theta) = \alpha\left(g - \frac{\lambda}{\alpha}\nabla_\theta K(\theta)\right). \tag{5}$$

In other words, the strength of the KL-penalty term is altered by a factor of $1/\alpha$ after normalization. We observe that unlike the unregularized scenarios, this attenuation cannot be recovered by changing other hyperparameters, such as learning rate: it represents a fundamental change to the objective.

Training Illustration. To clarify the effect this modification has on training, we simulate a training run over an objective $J(\theta) = \mathbb{E}_{x\sim\pi_\theta}[R(x)] - \lambda \cdot K(\theta)$, where the task is simple enough we can compute precise gradients for for the reward objective $\nabla_\theta\mathbb{E}_{x\sim\pi_\theta}[R(x)]$ and the KL-penalty $\nabla_\theta\lambda K(\theta)$ directly (without sampling) ($K(\theta) = D_{\mathrm{KL}}(\pi_\theta\|\pi_{\theta_0})$ and θ_0 denotes the initial policy parameters). We compute the gradient a scale factor would induce by applying the $1/\alpha$ multiplication to the KL-gradient and simulate training by applying the gradient update over multiple timesteps while varying α from $1/16$ to 64; details are in Section D.1.

Figure 1 tracks how the modified (panel (a)) and the original (panel (b)) objectives evolve throughout training. The contrast illustrates the effect of scale distortion: the modified objective rises as training proceeds, but for $\alpha > 1$, the original objective *falls* after peaking at around step 70 as the model implicitly optimizes for an attenuated KL penalty. For $\alpha < 1$ the original objective does not fall, but training plateaus at a much lower reward, being constrained by a much stronger KL penalty. We observe that for the $\alpha > 1$ runs in particular, a monitor tracking the modified objective (e.g., by reusing training samples from q_θ) could suggest that training is progressing successfully and hide that the true objective is decreasing silently. On the other hand, we also note that deviation from the original objective does not always mean bad performance: the $\alpha > 1$ cases did achieve a higher raw reward, which may be a desirable tradeoff against the increased KL-deviation in real-world scenarios.

2.2.2 ROTATION: CHANGING WHICH SOLUTIONS ARE REINFORCED

The scale-only case requires amplifying all nonzero rewards equally. In practice, enrichment will likely prefer a certain family of solutions over another (Shen et al., 2026b); the reweighting in Section 2.1 shows this leads to the reward of preferred samples being *relatively* amplified compared to those that are not. This relativity manifests as a *rotation* in the gradient space: rotated gradients follow different trajectories and can become trapped in (or escape from) different suboptimal policies.

Training Illustration. We illustrate the local minima effect using a similar training setup as in Section 2.2.1, where this time, the objective is unregularized and tracks raw expected reward $\mathbb{E}_{x\sim\pi_\theta}[R(x)]$. In addition to the base policy π_θ being trained, we construct an enriched policy q_θ from θ that prefers 0.5-reward samples but avoids 1-reward samples. We then compute the precise gradients g and u exactly, normalize both gradients to unit Euclidean norm, and perform a simulated training run as in Section 2.2.1. The training settings create an initial angle between g and u of roughly 39 degrees, eventually falling to roughly 7 degrees. Details can be found in Section D.1.

Figure 2 tracks the true objective $\mathbb{E}_{x\sim\pi_\theta}[R(x)]$ across training using the unit-normalized corrected (g) and uncorrected (u) gradients. With update lengths equalized, uncorrected training plateaus at reward 0.5, producing the 0.5-reward strings preferred by enrichment. Unenriched training, on the other hand, eventually reaches a policy 1-reward samples. The relative amplification explained in this section predicts exactly this phenomenon: in our setup, the enriched policy q_θ prefers the 0.5-reward samples by 4.3 times more than the 1-reward samples. This preference translates into an effective reward of 2.15 for 0.5-reward samples versus 0.46 for 1-reward samples steering uncorrected training towards the suboptimal 0.5-samples instead. This phenomenon is *not* a local minima that can be easily escaped; if, e.g., q_θ consistently prefers the 0.5-reward samples throughout the training trajectory, then the 0.5-sample rewards will always be amplified and training will consistently pull towards these samples. On the other hand, one can also see how an enrichment rotated towards high-reward samples would be highly useful in avoiding low-reward local minima during training, by amplifying their reward.

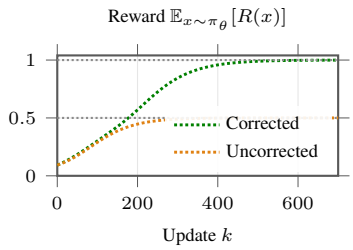


Figure 2: Reward evolution across update steps for uncorrected (orange) and corrected enriched (green) training.

2.2.3 VARIANCE: THE COST OF RESTORING THE INTENDED UPDATE

The illustrative experiments in Sections 2.2.1 and 2.2.2 give concrete examples of how training can be negatively affected due to uncorrected gradients. Although they do *not guarantee* suboptimality (indeed, distortion from enrichment can be beneficial as we will see in Section 4), it stands to reason that enriched, corrected gradients are also worth consideration. However, as discussed in Section 1, standard weight correction comes at a high cost of *variance* when estimating g with a corrected finite estimator $\hat{g}$, which can have its own undesirable consequences for training.The key behavior is when a rewarding sample x is *undersampled* in the enriched policy q_θ, compared to the base policy π_θ (i.e., the correction weight $(\pi_\theta(x))/q_\theta(x) > 1$). When this ratio is large, a finite batch during training may either (*i*) miss x altogether, reinforcing other rewarding samples preferred by q_θ, or (*ii*) the large corrected weight of x can dominate the batch and produce an abrupt parameter change; training can therefore depend strongly on if and when these two cases happen.

To make these two cases concrete, fix one rewarding response x that has a small but positive probability under q_θ, and let $\hat{g}$ average the importance-corrected contributions from N independent samples $x_1, \ldots, x_N \sim q_\theta$. If the batch does not contain x, it receives none of the gradient contribution associated with x; this happens with probability $\mathbb{P}(\forall i, 1 \le i \le N.\, x_i \neq x) = \big(1 - q_\theta(x)\big)^N$; when $N \cdot q_\theta(x) \ll 1$, most batches are of this kind.

On the other hand, a single occurrence of x in the batch contributes

$$\frac{1}{N}\frac{\pi_\theta(x)}{q_\theta(x)} R(x) \nabla_\theta \log \pi_\theta(x) \tag{6}$$

To the estimated gradient $\hat{g}$ (k occurrences in a batch contribute this term k times). In expectation, the contribution in Equation (6) averages to the desired $\pi_\theta(x)R(x)\nabla_\theta \log \pi_\theta(x)$, but the discrepancy

lies in the fact that a single occurrence of x supplies $1/(N \cdot q_\theta(x))$ times this average contribution to the gradient. Thus when $N \cdot q_\theta(x) \ll 1$ (which LLMs-as-policies are particularly prone to), the contribution becomes very large and the gradient can also entail an abrupt parameter change dominated by x. We observe that it is not just high variance in the gradients, but this particular bimodal manifestation, that becomes undesirable for training and is thus avoided in the literature (Li et al., 2026a; Zhang et al., 2026b;a; 2025) under the umbrella term 'variance'.

Training Illustration.

We illustrate the effects of variance through another controlled experiment where the enriched policy q_θ prefers 0.5-reward samples strongly compared to 1-reward samples, as in Section 2.2.2. We start from an initial base policy π_θ that generates 1-reward samples with probability ≈ 0.6; q_θ instead prefers 0.5-reward samples and gives 1-reward samples only $\approx 3.0 \cdot 10^{-5}$ probability: i.e., q_θ severely undersamples 1-reward samples.

We perform two types of training: base REINFORCE using samples from π_θ, and *corrected* (via weight ratios) enriched REINFORCE using samples from q_θ: the correction makes the expected update equivalent for both approaches, but only the latter configuration shows the bimodal variance we discussed. We vary the batch size N and perform 256 training runs for each configuration (N, policy), measuring the number of runs failing to reach a policy where reward ≥ 0.9.

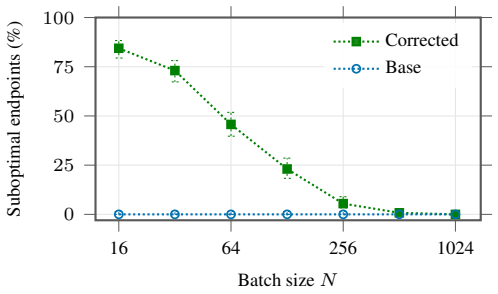


Figure 3: Percentage of reward < 0.9 endpoints generated by (*i*) corrected enriched and (*ii*) base REINFORCE, mapped against batch size.

Figure 3 illustrates this percentage mapped across batch sizes: the graph shows that when batch size $N = 16$, over 80% of policies end with reward < 0.9 under the corrected configuration. This effect is alleviated as N (and $N \cdot q_\theta(x)$) grows larger, but still visible at $N = 256$ where 5.5% of runs finish with reward < 0.9. Base REINFORCE avoids this problem; all runs succeed in reaching policies with reward ≥ 0.9 regardless of batch size, *even though* the expected update for both approaches are equivalent. This shows precisely how bimodal variance from weight correction can be harmful, and why existing work (Li et al., 2026a; Zhang et al., 2026b;a; 2025) refrains from applying correction.

3 Correcting Enriched Rollouts with Sequential Monte Carlo

Section 2.2.3 illustrates why bimodal variance can make weight correction undesirable for RL. In addition, we observe that for modern LLMs-as-policies, accumulated weights can be so large that computations on them cannot be represented stably in float32, making weight correction impractical.

The main contributor is that the correction weight $\pi_\theta(x)/q_\theta(x)$ accumulates *multiplicatively* as the length of the sample increases. For $x = t_1 \cdots t_l$, we have that $\frac{\pi_\theta(x)}{q_\theta(x)} = \prod_{j=1}^{l} \frac{\pi_\theta(t_j|t_{1:j-1})}{q_\theta(t_j|t_{1:j-1})}$: i.e., the total correction weight is the *product* of the per-token weight ratios returned by the LLM at each position. If the enriched policy consistently over- or undersamples the sample x at token-level granularity (which is relatively common in real LLM policies (Zhang et al., 2026c)), the total correction weight becomes exponential in the length of the sample. Verl (Sheng et al., 2025), an RL framework, thus internally enforces an log-upper bound of 20 on the size of the correction weight as a conservative guard to ensure float-32 representability for downstream computations. Our preliminary experiments from Section 4 often triggered this bound, indicating that naive weight correction is impractical, even if one is willing to accept the variance cost.

Thus to consider weight correction in any practical scenario, one must have some method of performing weight correction while keeping variance low. This section satisfies this goal by formulating a variant of sequential Monte-Carlo sampling (SMC) (Zhao et al., 2024) to draw samples from the enriched policy q_θ while reducing the variance cost of correction. In particular, SMC-style resampling tempers the multiplicative accumulation of correction weights, with an additive-in-length variance bound under the stability and particle-order assumptions in Section B.2.

3.1 Reducing Gradient Variance via SMC Resampling

Let us assume we are generating all N samples for a batch-gradient in parallel lockstep (i.e., we draw the first token for all N samples, then the second token for all N samples, and so on instead of generating each sample sequentially). For each sample x_i, during generation, we first attach x_i with a *correction weight* W_i that gets updated incrementally as each token gets drawn: $W_i = W_i \cdot \frac{\pi_\theta(t_k|t_{1:k-1})}{q_\theta(t_k|t_{1:k-1})}$ and the initial $W_i = 1$. [1] We observe that generating with W_i attached without any intervention will eventually yield $W_i = \pi_\theta(x_i)/q_\theta(x_i)$, the full correction weight, for each x_i.

A key intuition is that having incremental weights allows us to detect whether the variance in the final sample set is likely to become large *during* generation: by tracking the variance between W_i (i.e., the effective sample size (ESS) in terms of SMC (Doucet et al., 2001)). When this variance passes a certain threshold we trigger a *weighted resample* of the N current partial samples, proportional to their correction weights W_i, then *reset* each weight to the mean of W_i.

Example 2 (Variance Reduction)**.** Suppose $N = 4$ partial samples (x_1, x_2, x_3, x_4) have correction weights $(W_1, W_2, W_3, W_4) = (3, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, and one token remains to be generated. Let us assume that there are two possible final tokens, `t` and `u`, where at the final position their respective probabilities are $(1/4, 3/4)$ under π_θ and $(1/2, 1/2)$ under q_θ. Each final token therefore multiplies a correction ratio of either $1/2$ or $3/2$ (with equal probability); the variance of the ratio is $1/4$.

Without resampling, W_1 is either $3/2$ or $9/2$, while W_2, W_3, and W_4 are either $1/6$ or $1/2$, all with equal probabilities. On the other hand, with resampling, x_1 gains three copies in expectation while x_2, x_3, x_4 each gain $1/3$ each. The importance weights at this point are *reset* to the mean weight 1: thus after the final token is drawn, all samples have $W_i = 1/2$ or $3/2$. We observe that the expected variance of the correction weights in each scenario is:

without resampling:

$$\underbrace{\frac{1}{4^2}}_{\substack{\text{batch-mean}\\ \text{variance factor}}} \underbrace{\left(3^2 + 3\left(\frac{1}{3}\right)^2\right)}_{\substack{\text{sum of squared}\\ \text{attached weights}}} \underbrace{\frac{1}{4}}_{\substack{\text{variance of the}\\ \text{last-token ratio}}} = \frac{7}{48}$$

with resampling:

$$\underbrace{\frac{1}{4^2}}_{\substack{\text{batch-mean}\\ \text{variance factor}}} \underbrace{\left(1^2 + 1^2 + 1^2 + 1^2\right)}_{\substack{\text{sum of squared}\\ \text{reset weights}}} \underbrace{\frac{1}{4}}_{\substack{\text{variance of the}\\ \text{last-token ratio}}} = \frac{1}{16}.$$

Resampling has reduced this variance by a factor of $7/3$, while still preserving the expected contribution of each expected sample (because high-W_i samples generate more clones during resampling). The key reduction factor comes from the *remaining* variance induced by completing the partial samples. Without resampling, x_1's final weight depends on one final-token draw multiplied by 3, while after resampling, the contribution is spread across independent draws with weight 1, reducing variance of the contribution: this is the intuitive mechanism in how the SMC formulation tempers multiplicative variance accumulation into additive accumulation. Section D.2 provides a controlled experiment illustrating the variance reduction on gradients.

Once generation is complete, we use the final correction weights W_i to estimate the gradient with correction. The estimator $\hat{g}_{\text{SMC}}$ is then defined as and satisfies:

$$\hat{g}_{\text{SMC}} = \frac{1}{N}\sum_{i=1}^{N} W_i R(x_i) \nabla_\theta \log \pi_\theta(x_i), \qquad \mathbb{E}[\hat{g}_{\text{SMC}}] = g. \tag{7}$$

Resampling preserves the expected total correction weight of each prefix, which is the intuition in how SMC resampling preserves the mean of the expected gradient. We also note that the resampling mechanism induces a variance overhead, due to the fact that resampling copies prefixes (effectively reducing the number of distinct samples). Precise assumptions and a derivation of the additive-in-length bound, variance overhead, as well as a proof that $\mathbb{E}[\hat{g}_{\text{SMC}}] = g$, is provided in Section B.2. Section B.1 also provides an extension of the discussed SMC mechanism that that introduces a separate enrichment weight for better enrichment tracking.

One important qualification is that mean preservation as in Equation (7) does not automatically extend to GRPO, which can introduce both scale and rotation bias. However, we note that this distortion is not unique to SMC correction: even unenriched GRPO can induce a scale bias on individual problem gradients, which can further aggregate into a rotation bias when gradients are aggregated across

[1] We focus on per-token updates, but the idea transfers to coarser generation boundaries as well.

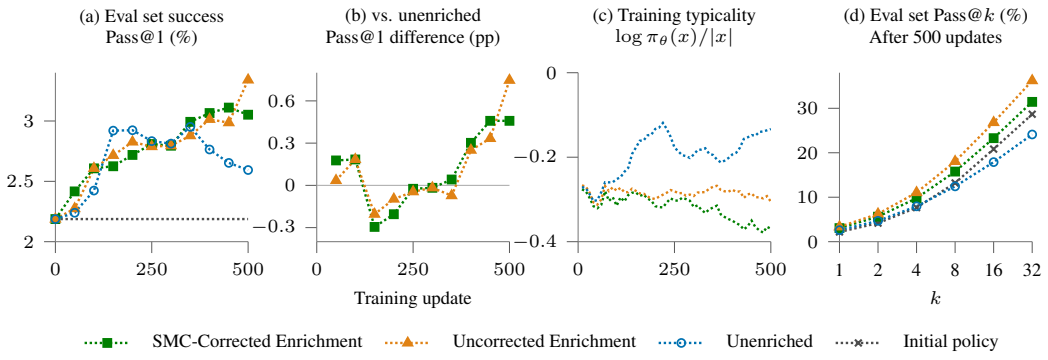


Figure 4: Results of training Qwen3-1.7B on the described OpenMathReasoning subset, using (*i*) uncorrected enrichment, (*ii*) SMC-corrected enrichment, and (*iii*) unenriched training. Panel (a) maps pass@1 percentages on the eval set across training steps. Panel (b) illustrates the advantage that uncorrected and SMC-corrected enrichment have compared to the unenriched baseline across training steps. Panel (c) maps training typicality, the average length-normalized likelihood of the training samples according to each policy being trained, across training steps. Panel (d) maps pass@k across k after all 500 training steps completed, illustrating that the two enriched arms achieve an increase compared to the base untrained policy while unenriched training results in a decrease.

problems (we illustrate this in Section A). Yao et al. (2026) provide a pairwise-surrogate justification of such group-relative updates; their weighted version applies to our own SMC-corrected reward update and establishes a principled practical use of SMC-correction within GRPO.

4 Uncorrected and Corrected Enrichment in Practice

We now connect our analysis to a real training run, by performing three different modes of RL to train Qwen3-1.7B (Yang et al., 2025b): (*i*) enriched and uncorrected, (*ii*) enriched and SMC-corrected, and (*iii*) unenriched base RL. Our goal is to compare the training trajectories of the three approaches, and analyze how the effect of enrichment we have established manifests itself in real training.

We train Qwen3-1.7B on a hard 500-problem subset of OpenMathReasoning where the pass@1 rate of the base policy is roughly 2%, retain a held-out 300-problem set for evaluation, and perform GRPO with a group size of 16 for 500 updates. Training results are averaged over three seeds. Our enrichment comes from a parallel LLM that is *hinted* with a partial solution, similar to (Agrawal et al., 2026), asked to produce a derivation and an answer after seeing the hint. We center our investigation on tracking policy performance across matched training steps.

Figure 4 illustrates the effects of enrichment. Panel (a) illustrates pass@1 performance evolving as training proceeds, showing that both enriched training approaches do achieve a better terminal policy compared to unenriched training. Panel (b) illustrates the difference between baseline unenriched training: at early training steps unenriched training outperforms both enriched trainers (roughly up to step 250), with the enriched trainers pulling ahead eventually by the terminal 500-th training step.

We explain this bimodal phenomenon via the analysis we have already established: at earlier steps, unenriched training sees a small amount of rewarding samples from the base policy and consistently reinforces the behavior on this set. This interpretation is supported by panel (c), which maps the typicality (length-normalized average log-probability) of the training samples a policy p_θ sees, according to p_θ itself, across training timesteps. Typicality grows until roughly 200 timesteps before plateauing, indicating that the unenriched-trained policy finds its own training rollouts more likely compared to its starting point. Panel (d) then shows that pass@k rate has *decreased* compared to the base policy for unenriched training, illustrating classical RL collapse: the unenriched trainer's sparse successes are concentrated on a few problems, and RL reinforces this narrow band of successes such that learning seems to proceed in early training steps (panel (a)) but eventually collapses towards memorizing a set of known solutions (the pass@k decrease in panel (d)).

On the other hand, the collapse does *not* occur for the two enriched trainers: in panel (c), both uncorrected and SMC-corrected show a mild decrease in typicality. This decrease comes from the policy's own entropy increasing hinted enrichment feeds it a diverse range of successful samples that the base policy cannot generate: the policy is learning new successes. In particular, panel (d) indicates that even the SMC-corrected arm has successfully increased pass@k, indicating that exposing the model to enriched successes has helped *prevent* collapse (even though the corrected weights are small). We do note that the difference between SMC-corrected and unenriched training was statistically insignificant at a 95% sample bootstrap interval (Berg-Kirkpatrick et al., 2012) of $[-0.21\text{pp}, +1.12\text{pp}]$ (a one-sided test of (SMC $>$ unenriched) yields $p = 0.09$). On the other hand, the pass@k increase between SMC and unenriched was statistically significant, indicating that although SMC maybe improving slower due to the corrected gradients, it is definitely avoiding the mode of collapse displayed by unenriched training.

Conclusion. Enrichment can be helpful for real training, especially in very sparse domains, where enrichment provides access to *new* successes the base policy would have failed to generate otherwise. The gradient distortion of uncorrected consumption is not necessarily harmful in this regime, especially as the distortion points towards accelerating learning on successes not in the base policy. Corrected consumption also benefits from having access to the new successes, establishing a concrete example domain and analysis of when enrichment can help real-world RL.

5 Discussion and Related Work

Strategies for enrichment. Zhang et al. (2026b); Huang et al. (2025); Zhang et al. (2025) use expert prefixes or hints to make successful continuations more accessible during training, similar to the hinted approach we used in Section 4. PaTR (Li et al., 2026b) instead uses process feedback to branch from promising intermediate states and stop unproductive paths. Resampling methods, such as sequential resampling, similarly reallocate computation among partial trajectories, with auxiliary particle filters (Pitt & Shephard, 1999) and weighted ensemble sampling (Aristoff, 2018) providing precedents for selection weights. Twisted SMC for language models (Zhao et al., 2024) can also be understood, in general, as an enrichment strategy (where the enrichment is provided by the twist) to reliably produce a set of samples that satisfy a certain property (e.g., having a reward). We apply the idea of twisted SMC to estimate corrected gradients for training a policy from enriched rollouts.

Learning from selected successes. STaR (Zelikman et al., 2022) and ReST$^{\text{EM}}$ (Singh et al., 2023) perform fine-tuning on successful responses that the model generates instead of performing reinforcement learning. Similarly, ReST-MCTS* (Zhang et al., 2024) uses process-reward-guided tree search to collect training traces. These methods explicitly learn from selected examples, rather than requiring an unbiased estimate of the base-policy gradient. We observe that uncorrected consumption of enrichment is equivalent (under scale normalization) to fine-tuning on the successful samples of an enriched policy; a derivation is provided in Section C.

Importance correction. Espeholt et al. (2018) uses truncated importance weights to handle actor–learner policy mismatch, illustrating the longstanding trade-off between correction and variance. Off-Context GRPO (Agrawal et al., 2026) addresses the mismatch between guided generation and unguided training through token-level importance ratios. Active importance sampling for policy gradients (Papini et al., 2024) explicitly optimizes the sampling policy to reduce gradient variance. In LLM RL, importance correction has recently been revisited: Yao et al. (2026) show that group-relative REINFORCE is already off-policy, Luo et al. (2025) and Zhang et al. (2026c) study sequence versus token-level ratios, and Huang et al. (2026) control off-policy variance via effective sample size. Our focus is how enrichment and its correction change learning, and in Section 4 we connect our analysis to a real Qwen3-1.7B training run to explain the effects of enrichment across training.

6 Conclusion

This paper studies enrichment itself as a component of RL. We mathematically establish enrichment as an implicit reweighting of rewards, clarify the effect that this reweighting can have on training, introduce SMC-correction to variance in weight correction, and analyze the effects of enrichment on a real training run. An interesting line of future work would be to develop an adaptive RL mechanism that looks at online policy behavior to choose the training strategy, based on the results in this paper.

AI USE STATEMENT

In this work, we used generative AI tools to help formulate the theorems in proofs in Sections 2 and 3 and the Appendices, design, implement and analyze the experiments in Section 4, investigate related work in Section 5, and improve figures and writing. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. We do not believe that our work raises questions related to the Code of Ethics.

REPRODUCIBILITY STATEMENT

Our work primarily concerns a mathematical analysis of the effects of enrichment. The code used to run the experiment in Section 4 and our implementation of the SMC corrector cannot be released at the time of review due to corporate licensing issues, but we plan to open-source the code upon publication.

REFERENCES

Priyank Agrawal, Ankur Samanta, Shervin Ghasemlou, Boris Vidolov, Jalaj Bhandari, Kavosh Asadi, Daniel Jiang, and Aditya Modi. Off-context grpo: Learning to reason on hard problems using privileged information. *arXiv preprint arXiv:2607.19313*, 2026.

David Aristoff. Analysis and optimization of weighted ensemble sampling. *ESAIM: Mathematical Modelling and Numerical Analysis*, 52(4):1219–1238, 2018.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pp. 995–1005, 2012.

Nicolas Chopin. Central limit theorem for sequential monte carlo methods and its application to bayesian inference. 2004.

Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pp. 64–69. Ieee, 2005.

Arnaud Doucet, Nando de Freitas, and Neil J. Gordon (eds.). *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer, 2001. ISBN 978-1-4419-2887-0. doi: 10.1007/978-1-4757-3437-9. URL https://doi.org/10.1007/978-1-4757-3437-9.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Thomas Foster, Anya Sims, Johannes Forkel, Mattie Fellows, and Jakob Foerster. Learning to reason at the frontier of learnability. *arXiv preprint arXiv:2502.12272*, 2025.

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

Luke J Huang, Zhuoyang Zhang, Qinghao Hu, Shang Yang, and Song Han. Stable asynchrony: Variance-controlled off-policy rl for llms. *arXiv preprint arXiv:2602.17616*, 2026.

Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M Ponti, and Ivan Titov. Blending supervised and reinforcement fine-tuning with prefix sampling. *arXiv preprint arXiv:2507.01679*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Thanh-Long V Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in llm reinforcement learning via entropy-guided advantage shaping. In *International Conference on Learning Representations*, volume 2026, pp. 121956–121982, 2026.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. In *International Conference on Learning Representations*, volume 2026, pp. 144191–144215, 2026a.

Xintong Li, Sha Li, Yuwei Zhang, Changlong Yu, Rongmei Lin, Hongye Jin, Shuyi Guan, Xin Liu, Linwei Li, Qingyu Yin, et al. Process reward informed tree rollout for effective multi-turn rl. *arXiv preprint arXiv:2607.15610*, 2026b.

Hailong Luo, Bin Wu, Hongyong Jia, Qingqing Zhu, and Lianlei Shan. Llm-cot enhanced graph neural recommendation with harmonized group policy optimization. *arXiv preprint arXiv:2505.12396*, 2025.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Matteo Papini, Giorgio Manganini, Alberto Maria Metelli, and Marcello Restelli. Policy gradient with active importance sampling. *arXiv preprint arXiv:2405.05630*, 2024.

Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

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

Guobin Shen, Chenxiao Zhao, Xiang Cheng, Lei Huang, and Xing Yu. Vespo: Variational sequence-level soft policy optimization for stable off-policy llm training. *arXiv preprint arXiv:2602.10693*, 2026a.

Xingyu Shen, Huishuai Zhang, Peng Li, Yinchun Wang, and Dongyan Zhao. Boosting llm exploration via weak-model guidance in rlvr. *arXiv preprint arXiv:2608.27420*, 2026b.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 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. *arXiv preprint arXiv:2312.06585*, 2023.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Advances in Neural Information Processing Systems*, 38:78500–78525, 2026.

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

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

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

Chaorui Yao, Yanxi Chen, Yuchang Sun, Yushuo Chen, Wenhao Zhang, Xuchen Pan, Yaliang Li, and Bolin Ding. Group-relative reinforce is secretly an off-policy algorithm: Demystifying some myths about grpo and its friends. In *International Conference on Learning Representations*, volume 2026, pp. 66179–66203, 2026.

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

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

Feng Zhang, Zezhong Tan, Xinhong Ma, Ziqiang Dong, Xi Leng, Jianfei Zhao, Xin Sun, and Yang Yang. Adhint: Adaptive hints with difficulty priors for reinforcement learning. *arXiv preprint arXiv:2512.13095*, 2025.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 131946–131974, 2026a.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *Advances in Neural Information Processing Systems*, 38:96726–96752, 2026b.

Yuheng Zhang, Chenlu Ye, Shuowei Jin, Changlong Yu, Wei Xiong, Saurabh Sahu, and Nan Jiang. Rethinking importance sampling in llm policy optimization: A cumulative token perspective. *arXiv preprint arXiv:2605.07331*, 2026c.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024.

A GRPO AND GRADIENT DISTORTION

In Section 2 we discussed how enrichment implicitly reweights rewards and distorts the gradient in a base REINFORCE setting. Here we extend the analysis to GRPO.

A.1 REWEIGHTED REWARD IN GRPO

Fix one problem and parameters θ, and draw $N \geq 2$ independent responses $x_1, \ldots, x_N \sim q_\theta$. We consider the response-level GRPO reward update at these rollout parameters, without clipping, length normalization, or KL regularization. Assume $\pi_\theta(x) > 0$ wherever $q_\theta(x) > 0$, and that the expectations below are finite. Define the group reward mean $\overline{R}$, sample standard deviation s_R, and advantages $\widehat{A}_i$ as in GRPO (Shao et al., 2024):

$$\overline{R} = \frac{1}{N}\sum_{j=1}^{N} R(x_j), \qquad s_R^2 = \frac{1}{N-1}\sum_{j=1}^{N}\big(R(x_j) - \overline{R}\big)^2, \qquad \widehat{A}_i = \frac{R(x_i) - \overline{R}}{s_R + \epsilon}, \tag{8}$$

where $\epsilon > 0$. Treating the advantages as fixed coefficients when differentiating, the uncorrected enriched update is

$$\widehat{u}_{\mathrm{GRPO}} = \frac{1}{N}\sum_{i=1}^{N} \widehat{A}_i \nabla_\theta \log \pi_\theta(x_i). \tag{9}$$

Each group position has the same expected contribution, so averaging over the full sampled group and then conditioning on x_1 gives

$$\begin{aligned}
\mathbb{E}_{x_1,\ldots,x_N \overset{\text{i.i.d.}}{\sim} q_\theta}[\widehat{u}_{\mathrm{GRPO}}] &= \mathbb{E}_{x_1,\ldots,x_N \overset{\text{i.i.d.}}{\sim} q_\theta}\Big[\widehat{A}_1 \nabla_\theta \log \pi_\theta(x_1)\Big] \\
&= \sum_x q_\theta(x)\, \mathbb{E}\Big[\widehat{A}_1 \mid x_1 = x\Big]\, \nabla_\theta \log \pi_\theta(x) \\
&= \mathbb{E}_{x\sim\pi_\theta}\Big[\underbrace{\frac{q_\theta(x)}{\pi_\theta(x)}\mathbb{E}\Big[\widehat{A}_1 \mid x_1 = x\Big]}_{R'^{\mathrm{GRPO}}_\theta(x)}\, \nabla_\theta \log \pi_\theta(x)\Big].
\end{aligned} \tag{10}$$

The conditional expectation averages over the remaining $N-1$ independent responses from q_θ. The outer sum averages over all possible values of x_1. Thus, the effective reward $R'^{\mathrm{GRPO}}_\theta$ can be understood as reweighting the expected group-relative advantage, as opposed to the raw reward R (similar to how GRPO replaces the raw reward with the group-relative advantage). As in Section 2.1, this reweighting factor depends on the current network parameters θ and is not fixed throughout training. The decomposition into scale, rotation, and variance, and their effects on training apply identically, but are realized under different conditions. Let $h = \mathbb{E}[\widehat{u}_{\mathrm{GRPO}}]$ and assume $g \neq 0$.

Scale. GRPO induces pure scale bias when the reweighted expected advantages satisfy:

$$R'^{\mathrm{GRPO}}_\theta(x) = \alpha R(x) + c \qquad \text{for every } x, \tag{11}$$

Here c is independent of the response x and $\alpha = \langle h, g\rangle / \|g\|^2$. Then $h = \alpha g$, since $\mathbb{E}_{x\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(x)] = 0$ eliminates the constant term.

Rotation. A sufficient condition for pure rotation is to transfer equal amounts of expected contribution between two responses whose score gradients differ but have the same projection onto g. Specifically, suppose that, for responses x_a, x_b, a nonzero δ, and a response-independent c,

$$R'^{\mathrm{GRPO}}_\theta(x) = R(x) + c + \delta\left(\frac{\mathbf{1}\{x = x_a\}}{\pi_\theta(x_a)} - \frac{\mathbf{1}\{x = x_b\}}{\pi_\theta(x_b)}\right). \tag{12}$$

Then the mean update is

$$h = g + \delta \nabla_\theta \log \frac{\pi_\theta(x_a)}{\pi_\theta(x_b)}. \tag{13}$$

If $\langle g, \nabla_\theta \log \pi_\theta(x_a)\rangle = \langle g, \nabla_\theta \log \pi_\theta(x_b)\rangle$ but the two score gradients differ, the added component is nonzero and perpendicular to g. Thus $\alpha = 1$ and $r \neq 0$: there is rotation error but no scale error in the decomposition of Section 2.2.

Variance. Consider binary rewards with success probability $p = \mathbb{P}_{x\sim q_\theta}(R(x) = 1) \in (0, 1)$. The probability that a group contains both successes and failures is $m = 1 - (1-p)^N - p^N$; every other group has identical rewards and hence zero advantages and zero update. Conditioning on whether the group is mixed gives

$$\mathbb{E}\big[\|\widehat{u}_{\mathrm{GRPO}} - h\|^2\big] \geq \frac{1-m}{m}\,\|h\|^2 . \tag{14}$$

For example, with $N = 128$ and $p = 10^{-4}$, approximately 98.7% of groups produce no update, and the variance is at least $77.6\,\|h\|^2$. For a nonzero mean h, this is high variance relative to the squared mean update: the expected learning signal is carried by rare mixed groups, rather than appearing consistently across updates.

A.2 GRPO Distorts Training Relative to Pure Reinforce

In addition to the reweighted reward derivation in Section A.1, here we note that GRPO updates distort training relative to pure REINFORCE training, *even without* enrichment. We derive the distortion by observing that GRPO induces a *per-problem* scale factor, which can further induce rotation distortion when aggregating gradients over multiple problems.

Set $q_\theta = \pi_\theta$ and assume binary rewards, with gradients computed by taking the sum over token log-probability gradients, multiplied by the corresponding group-relative advantage, and averaging over responses. For one problem, let $p_\theta = \mathbb{E}_{x\sim\pi_\theta}[R(x)] \in (0, 1)$ be its success probability, and write $\widehat{g}_{\mathrm{GRPO}}$ for the GRPO update from N independent base-policy responses. The true reference gradient is $g = \mathbb{E}_{x\sim\pi_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)]$.

Per-problem scale. Using $\mathbb{E}_{x\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(x)] = 0$, the conditional score means are

$$\begin{aligned}\mathbb{E}_{x\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(x) \mid R(x) = 1] &= \frac{g}{p_\theta},\\ \mathbb{E}_{x\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(x) \mid R(x) = 0] &= -\frac{g}{1-p_\theta}.\end{aligned} \tag{15}$$

Let $K = \sum_{i=1}^N R(x_i) \sim \mathrm{Binomial}(N, p_\theta)$ be the number of successes in the group. Conditional on $K = k$, the reward mean is k/N and the sample standard deviation is $s_k = \sqrt{k(N-k)/(N(N-1))}$. Substituting the positive and negative advantages gives

$$\begin{aligned}\mathbb{E}[\widehat{g}_{\mathrm{GRPO}} \mid K = k] &= \frac{k}{N}\frac{1-k/N}{s_k+\epsilon}\frac{g}{p_\theta} + \frac{N-k}{N}\frac{-k/N}{s_k+\epsilon}\frac{-g}{1-p_\theta}\\ &= \frac{k(N-k)}{N^2(s_k+\epsilon)p_\theta(1-p_\theta)}\,g.\end{aligned} \tag{16}$$

The all-failure and all-success groups ($k = 0, N$) contribute zero. Averaging over all groups therefore yields

$$\begin{aligned}\mathbb{E}_{x_1,\ldots,x_N \overset{\text{i.i.d.}}{\sim} \pi_\theta}[\widehat{g}_{\mathrm{GRPO}}] &= \alpha_N(p_\theta)\,g,\\ \alpha_N(p_\theta) &= \frac{1}{p_\theta(1-p_\theta)}\sum_{k=1}^{N-1}\binom{N}{k}p_\theta^k(1-p_\theta)^{N-k}\frac{k(N-k)}{N^2(s_k+\epsilon)}.\end{aligned} \tag{17}$$

The factor $\alpha_N(p_\theta)$ is positive but need not equal one: unenriched GRPO can introduce scale bias even on one problem, without rotating that problem's expected update.

Rotation across problems. Now consider two equally weighted problems at the same parameters θ, with success probabilities $p_{1,\theta}, p_{2,\theta}$ and true gradients g_1, g_2. The aggregate true gradient g_{all} and expected GRPO update h_{all} are

$$g_{\mathrm{all}} = \frac{g_1+g_2}{2}, \qquad h_{\mathrm{all}} = \frac{\alpha_N(p_{1,\theta})g_1 + \alpha_N(p_{2,\theta})g_2}{2}. \tag{18}$$

If g_1, g_2 are linearly independent and their scale factors differ, no single scalar maps g_{all} to h_{all}, so the aggregate update has rotation error. For example, with $N = 16$ and $\epsilon = 10^{-6}$, success probabilities 0.05 and 0.5 give scale factors approximately 3.16 and 1.87, respectively. Thus, per-problem normalization can change the relative contributions of problems and rotate their combined gradient, even without enrichment or importance correction.

B SMC CORRECTION

In this section we provide extra information and proofs about our SMC-corrector in Section 3.

B.1 DUAL WEIGHTS FOR ENFORCING ENRICHMENT

We observe that the mechanism presented in Section 3.1 reduces variance, but can counterbalance the effects of guidance. For example, if q_θ is a proper enrichment, then the correction weight W_i will be low for rewarding samples preferred by q_θ and high for non-rewarding samples preferred by π_θ: counteracting the enrichment at a resampling step. To alleviate this tension, we extend the presented mechanism by attaching an *additional enrichment weight* $B_i > 0$ to each sample x_i. The enrichment weight B_i tracks, intuitively, the expected future reward for x_i: for example, using a process reward model (PRM) score. We observe that B_i need not be tracked iteratively like W_i; it suffices that B_i is computable from the partial sample x_i. In this section, we assume that B_i is a suitable proxy for expected future reward; the SMC analogue for B_i is the resample potential Zhao et al. (2024).

We trigger resampling by tracking the variance of the product weight $W_i B_i$, where the weighted resample also uses the product weights for resampling: intuitively, multiplying the enrichment weight B_i restores the disadvantage that a sample x_i may have accumulated in W_i due to a proper enrichment. At a resample, a child sample x_c originating from x_i then inherits:

$$W_c = \frac{\sum_j W_j B_j}{N} \cdot \frac{1}{B_i} \qquad B_c = B_i. \tag{19}$$

Here the average is over the product weights $W_j B_j$, rather than the correction weights alone. The enrichment weight B_i is preserved because children inherit the same prefix. The intuition is that the enrichment weight increases the importance of the sample x_i by a factor of B_i, which results in more clones; thus the children of x_i after resampling must have a lower correction weight W to account for the additional samples. Once generation is complete, we discard the enrichment weights B_i and use only the correction weights W_i to compute the gradient: after all N samples are generated, $\hat{g}_{\mathrm{SMC}}$, the dual-weight SMC gradient estimator, is defined as:

$$\hat{g}_{\mathrm{SMC}} = \frac{1}{N} \sum_{i=1}^{N} W_i R(x_i) \nabla_\theta \log \pi_\theta(x_i) \tag{20}$$

B.2 MEAN PRESERVATION AND CONDITIONS FOR ADDITIVE VARIANCE ACCUMULATION

We prove that the SMC-corrected REINFORCE estimator preserves the true gradient in expectation, then derive conditions for an additive-in-length variance bound. Mean preservation does not require these additional variance-control conditions. In particular, an additive decomposition of the error does not by itself prevent its terms from growing exponentially with response length.

We first isolate the effect of resampling on correction weights, then derive a finite-population bound for the gradient estimator in Equation (20). We observe that the steps must be separate, because standard weight correction only does not account for the fact that two separate samples inside an SMC-sample batch can share the same prefix ancestry. We analyze the dual-weight rule in Section 3 with ESS-triggered systematic resampling, using conditional-variance accounting for particle estimators (Chopin, 2004).

Assumptions. Let us fix one problem and parameters θ during rollout, with a finite vocabulary and response length bounded by l. Assume finished responses are padded by a token of probability 1 under both policies, with correction ratio 1 and zero score gradient. All N particles start at the empty prefix with correction weight $W_i = 1$. The base and enriched policies have common positive support, and all importance ratios are exact, without clipping or self-normalization. We use three bounds, each controlling a different part of the estimator. First, for a next token a following prefix s, define the token-level correction ratio $\omega_\theta(a \mid s)$ and assume that its variance under the enriched policy is at most v:

$$\omega_\theta(a \mid s) := \frac{\pi_\theta(a \mid s)}{q_\theta(a \mid s)}, \qquad \mathrm{Var}_{a \sim q_\theta(\cdot \mid s)}[\omega_\theta(a \mid s)] \le v. \tag{21a}$$

The ratio has conditional mean 1 by normalization of π_θ; the assumption bounds the variability introduced by one token, not the variance of a full response's correction weight. The same constant v must apply at every reachable prefix.

Second, assume that the enrichment weight $B(s)$ is bounded above and bounded away from zero, and define ρ as the ratio of these bounds:

$$0 < b_{\min} \le B(s) \le b_{\max} < \infty, \qquad \rho := \frac{b_{\max}}{b_{\min}}. \tag{21b}$$

The positive lower bound makes division by $B(s)$ in the resampling reset well-defined, while ρ controls how much enrichment weights can differ across prefixes.

Third, assume there is a finite constant H bounding the norm of the reward-weighted score gradient for every complete response x:

$$\|R(x)\nabla_\theta \log \pi_\theta(x)\| \le H. \tag{21c}$$

This bounds the magnitude of a response's REINFORCE contribution before multiplying by its correction weight W. The bound H may depend on the response length l: the score is a sum of token-level gradients. We retain this dependence when interpreting the horizon-dependent variance bound below. These are sufficient conditions, not properties established for arbitrary enriched LLM policies. The full-gradient variance bound will additionally require the particle-order condition in Equation (33).

ESS-triggered weighted resampling. Following the token-level presentation in Section 3, we check ESS after each token except the last and resample only when

$$\mathrm{ESS}(WB) := \frac{(\sum_i W_i B_i)^2}{\sum_i (W_i B_i)^2} < \tau N \tag{22}$$

for a fixed trigger threshold $0 < \tau \le 1$. When resampling fires, we perform weighted resampling to select N parents, using normalized weights $p_i = W_i B_i / \sum_j W_j B_j$. If C_i counts the children of parent i, the resampling satisfies (Douc & Cappé, 2005)

$$\sum_i C_i = N, \qquad \mathbb{E}[C_i \mid \mathcal{F}] = N p_i, \qquad C_i \in \{\lfloor N p_i \rfloor, \lceil N p_i \rceil\}, \tag{23}$$

Here $\mathcal{F}$ denotes the sampling history before this resampling operation, including the current population and its order. Resampling uses fresh randomness conditional on this history; new token draws are conditionally independent across particles. The offspring counts need not be independent; the variance calculation below accounts for their dependence. Each child receives the reset in Equation (19).

After a skipped check, ESS is at least τN; after resampling, all child products $W_i B_i$ are equal, so ESS is N. Writing $\overline{W} = N^{-1} \sum_i W_i$, the bound on B therefore gives, before every generation operation,

$$\sum_i W_i^2 \le \frac{\sum_i (W_i B_i)^2}{b_{\min}^2} \le \frac{(\sum_i W_i B_i)^2}{b_{\min}^2 \tau N} \le \frac{\rho^2}{\tau} N \overline{W}^{\,2}. \tag{24}$$

Initialization also satisfies this inequality. Equation (24) is important, as it essentially bounds the variance on the weights after resampling, which we will use below.

Averaging before multiplying. First set $B = 1$ and retain the assumption that each token's correction ratio has conditional variance at most v. Each ratio has conditional mean 1 because $\sum_a q_\theta(a \mid s)\omega_\theta(a \mid s) = 1$, so its expected square is at most $1 + v$. Without resampling, a response's terminal weight W_i is the product of its l ratios, including the ratio-1 padding tokens for responses that finish early. Applying the bound successively at each token gives $\mathbb{E}[W_i^2] \le (1 + v)^l$. The expected squared weight $\mathbb{E}[W_i^2]$ is called its *second moment*; since $\mathbb{E}[W_i] = 1$, its variance is $\mathbb{E}[W_i^2] - 1$. For N independent responses, the variance of the average terminal weight is therefore bounded by

$$\mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N}\prod_{k=1}^{l}\omega_{\theta,k,i}\right] \le \frac{(1+v)^l - 1}{N}, \qquad \omega_{\theta,k,i} := \omega_\theta(t_{k,i} \mid t_{1:k-1,i}). \tag{25}$$

With ESS-triggered resampling, the weights need not be equal before each token, but Equation (24) bounds their concentration. Let $\overline{W}_k$ be the mean weight after token k and its optional resampling operation. When $B = 1$, resampling preserves the mean weight exactly, irrespective of which parents are selected. Conditioning on the population $\mathcal{F}$ before token k, independence of the new token draws gives

$$\mathbb{E}[\overline{W}_k^2 \mid \mathcal{F}] \le \overline{W}_{k-1}^2 + \frac{v}{N^2}\sum_i W_i^2 \le \left(1 + \frac{v}{\tau N}\right)\overline{W}_{k-1}^2. \tag{26}$$

Applying this inequality successively over the l token positions, starting from $\overline{W}_0 = 1$, bounds the expected squared terminal weight. Since its mean is 1, subtracting 1 gives the variance bound:

$$\mathrm{Var}[\overline{W}_l] \le \left(1 + \frac{v}{\tau N}\right)^l - 1 \le e^{lv/(\tau N)} - 1. \tag{27}$$

In the SMC bound, each new token's variance contribution is divided by at least τN *before* it compounds with earlier randomness (the $v/\tau N$ term in the middle equation), rather than averaging N already-compounded weights at the end. The upper bound is $lv/(\tau N) + O((lv/(\tau N))^2)$ when $lv/(\tau N)$ is small: its leading term accumulates additively over tokens. For example, when $lv \le \tau N$, the variance is at most $(e-1)lv/(\tau N)$. At fixed N, the bound remains exponential for sufficiently large l; the linear regime requires $lv/(\tau N)$ to remain bounded. Thus resampling reduces the rate of compounding in this bound, rather than removing exponential growth at every horizon.

Accounting for the full gradient. To convert the variance over correction weights into a full variance result over gradients, define the expected final gradient contribution from prefix s as

$$m_\theta(s) := \mathbb{E}_{x\sim\pi_\theta}\left[R(x)\nabla_\theta \log \pi_\theta(x) \mid x_{1:|s|} = s\right]. \tag{28}$$

Equation (28) satisfies $m_\theta(\emptyset) = g$ at the empty prefix, $m_\theta(x) = R(x)\nabla_\theta \log \pi_\theta(x)$ for a complete response, and $\|m_\theta(s)\| \le H$. For a current population of prefixes s_i, define its average correction weight $\overline{W}$, average selection weight $\overline{WB}$, and weighted expected gradient M by

$$\overline{W} = \frac{1}{N}\sum_i W_i, \qquad \overline{WB} = \frac{1}{N}\sum_i W_i B_i, \qquad M = \frac{1}{N}\sum_i W_i m_\theta(s_i). \tag{29}$$

We condition on the population $\mathcal{F}$ immediately before each operation, holding all prefixes s_i and weights W_i, B_i fixed. The remaining randomness is in the next-token draws for generation, or in the offspring counts C_i for resampling. Averaging over that randomness leaves M unchanged:

$$\begin{aligned} \sum_a q_\theta(a \mid s)\omega_\theta(a \mid s) m_\theta(sa) &= m_\theta(s), \\ \mathbb{E}[M^{\mathrm{res}} \mid \mathcal{F}] = \frac{\overline{WB}}{N}\sum_i \frac{\mathbb{E}[C_i \mid \mathcal{F}]}{B_i} m_\theta(s_i) &= M. \end{aligned} \tag{30}$$

Here M^{res} is the weighted expected gradient after resampling. The second equality uses expected offspring counts, not independent parent draws. A skipped resampling check leaves M unchanged, and the ESS decision is determined by the current population, so the adaptive schedule also preserves the conditional mean.

To accumulate the errors over a full rollout, number the l generation operations and $l-1$ resampling checks by $j = 1, \ldots, 2l-1$ in the order they occur. A skipped check still receives an index, but leaves the population unchanged. Write M_j for M evaluated on the population after operation j, and let $\mathcal{F}_j$ denote the sampling history up to that point. This history records the tokens drawn and parents selected so far, and therefore determines all current prefixes and weights. Conditioning on $\mathcal{F}_{j-1}$ means holding this history fixed and averaging only over the new random choices in operation j. The preceding mean-preservation identities then give $\mathbb{E}[M_j \mid \mathcal{F}_{j-1}] = M_{j-1}$: the change $M_j - M_{j-1}$ has conditional mean zero. Initially $M_0 = g$, and after the final generation operation $M_{2l-1} = \hat{g}_{\mathrm{SMC}}$. Taking expectations successively over these operations therefore gives

$$\mathbb{E}[\hat{g}_{\mathrm{SMC}}] = \mathbb{E}[M_{2l-1}] = \cdots = \mathbb{E}[M_0] = g.$$

Thus the SMC-corrected REINFORCE estimator is unbiased for the true gradient g, as stated in Section 3. This conclusion follows from mean preservation alone and does not require the variance bounds derived below.

This property also makes the cross terms between different operations vanish in expectation. For $j < k$, fixing the history $\mathcal{F}_{k-1}$ fixes the earlier change $M_j - M_{j-1}$, while the later change $M_k - M_{k-1}$ still has conditional mean zero. Their expected inner product is therefore zero: $\mathbb{E}[\langle M_j - M_{j-1}, M_k - M_{k-1}\rangle] = 0$. The final error $\hat{g}_{\mathrm{SMC}} - g$ is the sum of these changes, so expanding its squared norm and taking expectations gives

$$\mathbb{E}\left[\|\hat{g}_{\mathrm{SMC}} - g\|^2\right] = \sum_{j=1}^{2l-1} \mathbb{E}\left[\|M_j - M_{j-1}\|^2\right]. \tag{31}$$

This identity includes resampling noise and the dependence caused by shared ancestors. An additive decomposition alone does not imply linear growth: its individual terms must also remain controlled.

Bounding systematic resampling noise. The ESS bound controls the weights, but not the correlations between offspring counts. By Equation (23), each C_i takes at most two adjacent integer values, so $\mathrm{Var}(C_i \mid \mathcal{F}) \le 1/4$. For any real coefficients $z_1, \dots, z_N$, Cauchy–Schwarz therefore gives

$$\begin{aligned}\mathrm{Var}\left[\sum_i z_i C_i \mid \mathcal{F}\right] &= \mathbb{E}\left[\left(\sum_i z_i (C_i - N p_i)\right)^2 \mid \mathcal{F}\right] \\ &\le \left(\sum_i z_i^2\right) \sum_i \mathrm{Var}(C_i \mid \mathcal{F}) \le \frac{N}{4} \sum_i z_i^2.\end{aligned} \tag{32}$$

This bound includes correlations between offspring counts and does not assume independent parent selections. It need not provide a variance reduction as N grows: systematic resampling can correlate many offspring counts through its shared random offset (Douc & Cappé, 2005). For example, with even N, alternating values $Np_i = 3/2, 1/2$ and coefficients $z_i = 1, -1$ give variance $N^2/4$, attaining Equation (32). We therefore state an additional condition on the ordered population, rather than infer a $1/N$ factor from ESS alone.

Additional assumption for additive variance: controlled variation in particle order. At each triggered resample, index the parents $s_1, \dots, s_N$ in the order used by systematic resampling. Assume that a constant $K_{\mathrm{ord}} > 0$, independent of N, l, and the resampling time, bounds the total change between neighboring parents as follows:

$$\begin{aligned}\sum_{i=1}^{N-1} \left\| \frac{m_\theta(s_{i+1})}{B_{i+1}} - \frac{m_\theta(s_i)}{B_i} \right\| &\le 2\sqrt{K_{\mathrm{ord}} N}\, \frac{H}{b_{\min}}, \\ \sum_{i=1}^{N-1} \left| \frac{1}{B_{i+1}} - \frac{1}{B_i} \right| &\le \sqrt{K_{\mathrm{ord}} N} \left(\frac{1}{b_{\min}} - \frac{1}{b_{\max}} \right).\end{aligned} \tag{33}$$

The first inequality controls variation in the resampled gradient contributions; the second controls variation in the resampled total correction weight. They allow total variation to grow as $\sqrt{N}$, but exclude repeated fixed-amplitude alternation such as the counterexample above. For example, in one dimension, if both coefficient sequences are monotone, their total variations are at most $2H/b_{\min}$ and $1/b_{\min} - 1/b_{\max}$, respectively, so $K_{\mathrm{ord}} = 1$ suffices. The reciprocal-weight condition is automatic when $B = 1$.

To see their effect, let D_k be the discrepancy between the total offspring of the first k parents and its conditional expectation:

$$D_k := \sum_{i=1}^{k} (C_i - N p_i), \qquad D_N = 0, \qquad \mathbb{E}[D_k \mid \mathcal{F}] = 0, \qquad \mathbb{E}[D_k^2 \mid \mathcal{F}] \le \frac{1}{4}. \tag{34}$$

The last bound follows because systematic resampling rounds each cumulative offspring count to one of the two integers surrounding $N \sum_{i=1}^{k} p_i$. For any fixed vector coefficients z_i, summation by parts gives $\sum_i (C_i - N p_i) z_i = \sum_{k=1}^{N-1} D_k (z_k - z_{k+1})$. The triangle inequality for the conditional root-mean-square norm therefore yields

$$\mathbb{E}\left[\left\| \sum_i (C_i - N p_i) z_i \right\|^2 \mid \mathcal{F} \right] \le \frac{1}{4} \left(\sum_{i=1}^{N-1} \|z_{i+1} - z_i\| \right)^2. \tag{35}$$

This bound retains the dependence between offspring counts; no independence assumption is introduced.

Bounding the individual errors. Write M^{gen} for the weighted expected gradient after a generation operation. Conditional independence of token draws and Equation (24) give

$$\mathbb{E}\Big[\|M^{\mathrm{gen}} - M\|^2 \mid \mathcal{F} \Big] \le \frac{(1+v) H^2}{N^2} \sum_i W_i^2 \le \frac{\rho^2 (1+v) H^2}{\tau N} \overline{W}^2. \tag{36}$$

For a triggered systematic resample, applying Equations (33) and (35) yields

$$\begin{aligned}\mathbb{E}\Big[\|M^{\mathrm{res}} - M\|^2 \mid \mathcal{F} \Big] &= \frac{\overline{WB}^2}{N^2} \mathbb{E}\left[\left\| \sum_i (C_i - N p_i) \frac{m_\theta(s_i)}{B_i} \right\|^2 \mid \mathcal{F} \right] \\ &\le \frac{K_{\mathrm{ord}} H^2}{N b_{\min}^2} \overline{WB}^2 \le \frac{K_{\mathrm{ord}} \rho^2 H^2}{N} \overline{W}^2.\end{aligned} \tag{37}$$

The final inequality uses $\overline{WB} \le b_{\max}\overline{W}$. A skipped check has zero resampling variance and satisfies the same upper bound. The remaining factor $\overline{W}^2$ is random and must not be replaced by 1. Repeating the same calculation with the constant function 1 in place of m_θ yields

$$\begin{aligned}\mathbb{E}[(\overline{W}^{\mathrm{gen}})^2 \mid \mathcal{F}] &\le \left(1 + \frac{\rho^2 v}{\tau N}\right)\overline{W}^2,\\ \mathbb{E}[(\overline{W}^{\mathrm{res}})^2 \mid \mathcal{F}] &\le \left(1 + \frac{K_{\mathrm{ord}}(\rho-1)^2}{4N}\right)\overline{W}^2.\end{aligned} \tag{38}$$

For the second line, apply Equation (35) to $z_i = 1/B_i$ and use the second inequality in Equation (33). This bounds the conditional variance added by resampling by $K_{\mathrm{ord}}(\rho-1)^2\overline{W}^2/(4N)$; the conditional mean remains $\overline{W}$. In particular, this resampling contribution vanishes when $B = 1$, as in the weight-only calculation above. Let $\overline{W}_k$ denote the mean weight after token position k and its resampling check, if any. Starting from $\overline{W}_0 = 1$, applying these bounds successively over the first k token positions gives

$$\mathbb{E}[\overline{W}_k^2] \le \left[\left(1 + \frac{\rho^2 v}{\tau N}\right)\left(1 + \frac{K_{\mathrm{ord}}(\rho-1)^2}{4N}\right)\right]^k \le e^{\frac{k}{N}\left(\rho^2 v/\tau + K_{\mathrm{ord}}(\rho-1)^2/4\right)}. \tag{39}$$

The same upper bound applies just before resampling at token position k. Substituting these bounds into Equation (31), with l generation terms and at most $l-1$ nonzero resampling terms, proves

$$\mathbb{E}\left[\|\hat{g}_{\mathrm{SMC}} - g\|^2\right] \le \frac{\rho^2 H^2 l}{N}\left(\frac{1+v}{\tau} + K_{\mathrm{ord}}\right) e^{\frac{l}{N}\left(\rho^2 v/\tau + K_{\mathrm{ord}}(\rho-1)^2/4\right)}. \tag{40}$$

The explicit factor l comes from adding the errors of the l generation steps and at most $l-1$ resampling steps. ESS controls the generation terms, while the particle-order assumption controls the systematic-resampling terms.

Horizon regime and scope of the bound. For the additive-in-length conclusion, assume that ρ, v, $1/\tau$, and K_{ord} remain bounded and that, for a constant c independent of l and N,

$$\frac{l}{N}\left(\frac{\rho^2 v}{\tau} + \frac{K_{\mathrm{ord}}(\rho-1)^2}{4}\right) \le c. \tag{41}$$

Then Equation (40) is $O(H^2 l/N)$, rather than the $H^2(1+v)^l/N$ upper bound obtained for independent importance-corrected gradients from $\|R\nabla_\theta \log \pi_\theta\| \le H$ and $\mathbb{E}[W^2] \le (1+v)^l$. We observe that if l grows very large and N is fixed, the SMC bound can still grow exponentially outside Equation (41). Because the factor is $O(H^2 l/N)$, one also requires that H remains bounded across l for a true linear bound.

For comparison, $\|\hat{g}_{\mathrm{SMC}}\| \le H\overline{W}_l$ and unbiasedness also give the direct bound

$$\mathbb{E}\left[\|\hat{g}_{\mathrm{SMC}} - g\|^2\right] \le H^2\mathbb{E}[\overline{W}_l^2] - \|g\|^2. \tag{42}$$

One may take the smaller of this bound and Equation (40); the latter exposes the l/N factor when the accumulated error is small.

C Uncorrected Enrichment as Success-Filtered SFT

In this section, we characterize RL over binary rewards using an uncorrected enrichment q_θ as equivalent up to a scale factor to supervised-fine tuning (SFT) on samples from q_θ that have reward 1.

Fix one problem and binary terminal rewards $R(x) \in \{0, 1\}$. We consider baseline-free REINFORCE, summing token log-probability gradients and averaging over responses, without clipping or KL regularization. Let $\mu_\theta^q = \mathbb{P}_{x\sim q_\theta}(R(x) = 1) > 0$ be the enriched policy's success probability. Filtering its responses to retain only successes gives the distribution

$$q_\theta^+(x) = q_\theta(x \mid R(x) = 1) = \frac{q_\theta(x)R(x)}{\mu_\theta^q}. \tag{43}$$

The expected uncorrected update therefore satisfies

$$\begin{aligned}u &= \mathbb{E}_{x\sim q_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)]\\ &= \mu_\theta^q\, \mathbb{E}_{x\sim q_\theta^+}[\nabla_\theta \log \pi_\theta(x)].\end{aligned} \tag{44}$$

The remaining expectation is the SFT ascent direction on successful enriched responses. To make this explicit, let ϑ denote the parameters being optimized while θ denotes the fixed rollout parameters, and define the SFT loss $\mathcal{L}_{\mathrm{SFT}}(\vartheta;\theta)$ by

$$\mathcal{L}_{\mathrm{SFT}}(\vartheta;\theta) = -\mathbb{E}_{x\sim q_\theta^+}[\log \pi_\vartheta(x)]\,, \qquad u = -\mu_\theta^q\, \nabla_\vartheta \mathcal{L}_{\mathrm{SFT}}(\vartheta;\theta)|_{\vartheta=\theta}\,. \tag{45}$$

Thus, uncorrected enriched REINFORCE follows the success-filtered SFT gradient, multiplied by the enriched success probability. In this sense, uncorrected enrichment trains the base policy to imitate the successful responses that enrichment produces, just like SFT.

Batch-level equivalence. For a batch of N enriched responses, let $K = \sum_{i=1}^N R(x_i)$ be the number of successes. Whenever $K > 0$, the same identity holds for the realized batch:

$$\underbrace{\frac{1}{N}\sum_{i=1}^N R(x_i)\nabla_\theta \log \pi_\theta(x_i)}_{\text{uncorrected REINFORCE}} = \frac{K}{N}\underbrace{\frac{1}{K}\sum_{i:R(x_i)=1} \nabla_\theta \log \pi_\theta(x_i)}_{\text{success-filtered SFT ascent}}\,. \tag{46}$$

When $K = 0$, REINFORCE produces zero update and there are no successful responses for SFT. The difference is normalization: REINFORCE averages over all responses, whereas success-filtered SFT averages over successes. For one problem this changes only the gradient magnitude; if SFT averages successes separately within each problem and then weights problems equally, omitting the success-rate factors can also change the aggregate gradient direction (similar to how cross-problem gradient aggregation can change the direction of GRPO gradients, as in Section A).

D Training Setups and Extra Experiments

We provide detailed descriptions for the controlled experiments in Section 2, as well as an additional controlled experiment illustrating the variance reduction of SMC from Section 3.

D.1 Training Setups for Experiments in Section 2

Policy architecture. The experiments in Sections 2.2.1 to 2.2.3 use a tabular autoregressive policy: a lookup table followed by a softmax, with no hidden layers. Its input is the sequence of tokens generated so far (the prefix s), and its output is a distribution over the next token $a \in \{0, 1, 2\}$. Each prefix has its own row of three trainable logits $\theta_{s,a}$, giving $\pi_\theta(a \mid s) = e^{\theta_{s,a}} / \sum_{b=0}^2 e^{\theta_{s,b}}$; parameters are not shared across prefixes. Generation starts from the empty prefix, appends one token at a time, and stops after four tokens. There are therefore $1 + 3 + 9 + 27 = 40$ possible prefixes at which the model makes a decision, giving 120 trainable parameters.

Task. The policy is trained to generate sequences of length 4 that receive high reward under a fixed scoring rule. Reward is terminal, given only on the full sample. A complete sequence x receives $R(x) = 1$ if it begins with `000`, $1/2$ if it begins with `11`, and 0 otherwise. For example, `0002` earns 1, `1102` earns $1/2$, and `0102` earns 0. Of the $3^4 = 81$ possible sequences, three receive reward 1 and nine receive reward $1/2$; the variance experiment changes the reward-1 prefix to `00`, as specified below.

Shared training. Unless stated otherwise, all logits start at zero, assigning probability $1/3$ to each next token and $1/81$ to each complete sequence. Updates use the base-policy score $\nabla_\theta \log \pi_\theta(x)$, without reward baselines, GRPO normalization, momentum, adaptive optimization, or gradient clipping. Evaluation uses exact base-policy expected reward, not sampled estimates.

Scale. We use the objective $J(\theta) = \mathbb{E}_{x\sim\pi_\theta}[R(x)] - 0.4K(\theta)$, where $K(\theta) = D_{\mathrm{KL}}(\pi_\theta \| \pi_{\theta_0})$ is computed over complete responses relative to the fixed initial policy. For each $\alpha \in \{1/16, 1/4, 1, 4, 16, 64\}$, we multiply the reward gradient by α, leave the KL gradient unchanged, and divide the learning rate by α. The resulting update is

$$\theta_{k+1} = \theta_k + 0.5\, \nabla_\theta \left(\mathbb{E}_{x\sim\pi_\theta}[R(x)] - \frac{0.4}{\alpha}K(\theta)\right)\Big|_{\theta=\theta_k} \tag{47}$$

The coefficient 0.5 is the learning rate. We compute both gradients exactly and run 400 updates per value of α to simulate a training run with a scale-only difference, as stated in Section 2.2.1.

Rotation. We set $J(\theta) = \mathbb{E}_{x\sim\pi_\theta}[R(x)]$ and construct the enriched policy q_θ by multiplying the base probability of token 1 by $e^{1.5}$ at the empty prefix and at prefix `1`, then normalizing the three action weights at each prefix. All other next-token distributions are unchanged, so the guide favors completing `11` without assisting the reward-1 family. We compute $g = \mathbb{E}_{x\sim\pi_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)]$ and $u = \mathbb{E}_{x\sim q_\theta}[R(x)\nabla_\theta \log \pi_\theta(x)]$ exactly at each run's current parameters. Starting from the same uniform policy, the two runs take 700 updates in their respective scale-normalized directions, each with Euclidean step length 0.02. Thus neither sampling noise nor differences in update length explain the comparison.

Variance. We set $J(\theta) = \mathbb{E}_{x\sim\pi_\theta}[R(x)]$, and then change the reward such that prefixes with `00` (instead of `000`) receive reward 1. We then initialize the token-0 logits at the empty prefix and prefix `0` to 2, leaving all other logits at zero. The enriched policy q_θ retains the rotation experiment's $e^{1.5}$ preference toward `11` and additionally multiplies token-0 probabilities at the empty prefix and prefix `0` by e^{-6} before local normalization. The initial reward-1 family probabilities are consequently 0.6193 under π_{θ_0} and 3.022×10^{-5} under q_{θ_0}. For each $N \in \{16, 32, 64, 128, 256, 512, 1024\}$, we run 256 independent trials per N, each taking 12,670 updates with learning rate 0.05 and N fresh responses per update. The unenriched arm averages $R(x)\nabla_\theta \log \pi_\theta(x)$ for samples from π_θ; the corrected arm samples from q_θ and multiplies each contribution by $\pi_\theta(x)/q_\theta(x)$, without clipping or self-normalization, to compute the gradient. An endpoint is counted as suboptimal when its expected reward is below 0.9, and error bars are pointwise 95% confidence intervals. Both arms have equal sample budgets at each N; increasing N increases the total budget, rather than merely reorganizing a fixed number of samples.

D.2 Controlled Experiment Illustrating SMC Variance Reduction

Setup. We use a tabular autoregressive policy with 32 parameters: 16 softmax logits for selecting a target string and one match-probability logit per target. The first token selects one of 16 fixed target strings; the policy then generates l tokens from a vocabulary of size 16. At each position, the policy emits the corresponding target token with a target-specific probability and distributes the remaining probability uniformly over the other 15 tokens. A completed response receives reward 1 if its number of matching tokens exceeds a threshold and 0 if it falls below; at the threshold, an independent coin flip determines the binary reward. We set the threshold and coin probability so that the base success rate is exactly 10^{-5} at every length, separating the effect of horizon from changing reward sparsity. We use 24 fixed policy/task instances and vary $l \in \{24, 32, 48, 64, 80, 96, 112, 128\}$, excluding the initial selection token from l. Unlike the experiments above, we freeze the policy parameters and compare gradient estimates rather than training outcomes. To construct the enrichment, we compute $1 + (e^8 - 1)\mathbb{P}_{\pi_\theta}(R = 1 \mid s)$ at each prefix s, then perturb its logarithm with fixed random noise of root-mean-square magnitude 0.5 to obtain the enrichment weight $B(s)$. The noise depends on token position and match count, is shared across targets, and leaves the empty prefix and terminal weights unchanged. The enriched policy q_θ multiplies each next token's base probability by the enrichment weight of the resulting prefix and normalizes locally. Although this is a constructed domain to illustrate the effect of SMC variance reduction, we note that such random-noise perturbed policies can arise commonly in token-level proposals, where constant training makes the parameters of an LLM drift, and subsequently a perfectly cancelling enrichment (i.e., a perfect twist as in (Zhao et al., 2024)) cannot exist.

We compare $N = 512$ independent samples from π_θ, $N = 512$ independent samples from q_θ with full importance correction, and $N = 512$ particles using the dual-weight SMC corrector in Section B.1. SMC checks the ESS of the product weights $W_i B_i$ after each nonterminal token and resamples when it falls below $0.8N$. All three estimators subtract the same fixed reward baseline $J(\theta) = 10^{-5}$; correction weights are neither clipped nor self-normalized, preserving the expected gradient g. We compute g and the two independent estimators' variances exactly by dynamic programming over the selected target, token position, and match count. For SMC, we estimate variance as mean squared error relative to g over 64 independent batches per instance and length; MSE equals variance because the estimator is unbiased.

Results. Figure 5 shows that independent correction and SMC have similar variance at short horizons, but diverge sharply as the responses grow longer. The variance of independent correction is 1.13 times that of SMC at $l = 24$, rising to approximately 3,929 times at $l = 128$. At the longest

horizon, independent correction has approximately 35 times the variance of unenriched sampling, whereas SMC's variance is approximately $1/112$ that of unenriched sampling.

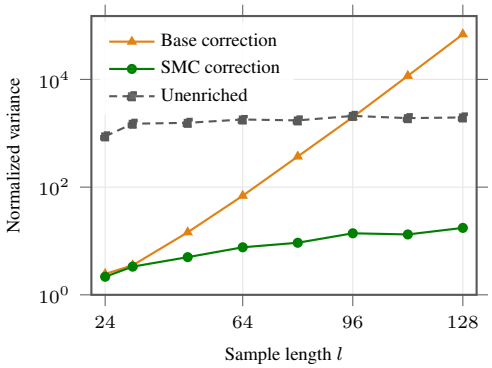


Figure 5: Normalized variance compared over enriched base correction (as described at the head of Section 3, SMC correction, and unenriched estimators.