REVO: ROLLOUT-EFFICIENT OFF-POLICY DISTILLATION VIA VARIANCE-GUIDED REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) trains language models using dense token-level teacher supervision on student-generated trajectories. However, its reliance on frequently refreshed student rollouts often incurs substantial generation cost. We introduce REVO, an off-policy distillation framework that improves rollout efficiency by reusing each student rollout for multi-step learner updates. REVO addresses prefix-level and current-token policy mismatch through stabilized prefix weighting and one-step resampling from the current student, which enables repeated updates without regenerating full trajectories. To prioritize informative token positions within reused rollouts, REVO uses the variance of the student-teacher log-probability ratio to quantify the remaining token-level learning signal and guide repeated optimization. Across multiple student–teacher scales, REVO with only 50 rollout iterations matches or exceeds OPD baselines trained for 200 iterations on both in-domain and cross-domain reasoning benchmarks.

1 INTRODUCTION

Learning from self-generated rollouts has emerged as a promising approach to language model post-training. Among these approaches, on-policy distillation (OPD) (Agarwal et al., 2024; Lu & Lab, 2025; Gu et al., 2024; Ko et al., 2025) leverages dense token-level supervision by querying a teacher model on student-generated prefixes. In particular, OPD aligns the student's next-token distribution with the teacher's at these prefixes by minimizing the token-level reverse-KL objective $\mathbb{E}_{a\sim P_\theta(\cdot|c_t)}[\log \frac{P_\theta(a|c_t)}{P_T(a|c_t)}]$, where c_t denotes a student-generated prefix and P_θ and P_T denote the student and teacher policies, respectively (Lu & Lab, 2025; Li et al., 2026a).

Despite this favorable optimization structure, OPD remains expensive because of the cost of collecting student generations. Although it often reaches strong performance within a few hundred iterations, each iteration requires a fresh batch of student rollouts, so even a relatively short training run can involve thousands of full autoregressive generations. Prior work has shown that rollout generation can dominate the wall-clock cost of OPD training (Ziheng et al., 2026). Reusing trajectories through *off-policy optimization* therefore provides a natural way to amortize this cost by performing multiple learner updates on existing student-generated contexts. However, effective off-policy distillation introduces challenges in both training stability and the quality of the learning signal extracted from reused data, as detailed below.

First, as the student evolves during training, the distributions of stored prefixes and their sampled tokens can diverge from those induced by the current policy, biasing naive data reuse. We distinguish two sources of this distribution shift: *prefix mismatch* and *current-token mismatch*. Correcting prefix mismatch with standard importance sampling requires multiplying token likelihood ratios along the prefix, which can produce highly variable weights and unstable updates for long trajectories. Current-token mismatch arises from retaining tokens sampled by an earlier student (Zhang et al., 2026; Kang et al., 2026). Unlike the prefix, these tokens can be resampled from the current student at the stored prefix without regenerating the trajectory.

Second, rollout reuse makes the strength of the remaining distillation signal particularly important. As observed by Jin et al. (2026) and Li et al. (2026b), the strength of this signal varies across token positions. Repeatedly revisiting the same prefixes can therefore devote additional computation to

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

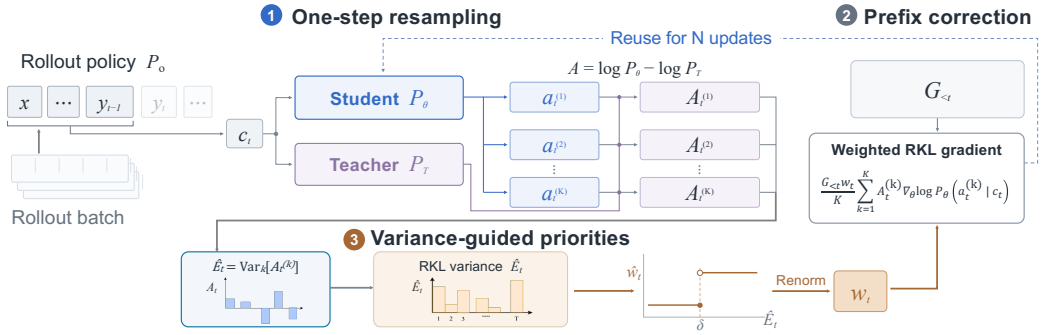

Figure 1: **Overview of REVO.** One-step resampling refreshes token-level teacher supervision at stored prefixes, while stabilized prefix weights and variance-guided token priorities support repeated updates.

positions with little remaining learning signal. Effective off-policy updates should prioritize informative token positions while limiting the influence of sampling noise.

Motivated by these challenges, we propose REVO, a rollout-efficient off-policy distillation framework that performs multiple learner updates on each student rollout under dense token-level teacher supervision, as outlined in Figure 1. REVO combines two complementary components: stabilizing updates on historical student rollouts and adaptively prioritizing their remaining distillation signal. Specifically, REVO introduces *one-step resampling* to draw fresh tokens from the current student at stored prefixes, removing the need for conditional current-token importance weighting. It further stabilizes off-policy updates with a geometrically normalized, capped prefix weight (Zheng et al., 2025). In parallel, REVO leverages the same resampled tokens for *variance-guided token prioritization*. We estimate the variance of the student and teacher log probability ratio and use it to assign greater weight to positions with stronger remaining learning signal and more reliable finite sample gradient estimates. Under a local realizability condition, we show that this variance equals the inner product between the ordinary and natural gradients of the local reverse-KL objective, connecting it to first-order natural-gradient progress. Empirically, this statistic is strongly associated with the signal-to-noise ratio of finite-sample logit-gradient estimates. Together, these components enable current-policy token supervision at reused off-policy prefixes and couple the refreshed supervision with adaptive prioritization, without additional model evaluations for variance estimation. Our contributions are threefold:

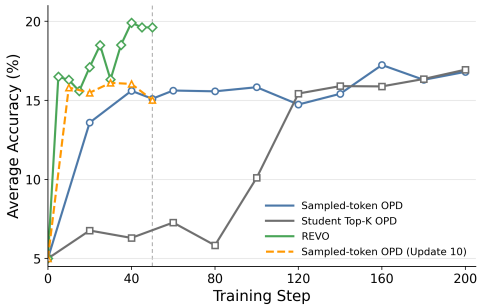

Figure 2: **Rollout efficiency of REVO.** For Qwen3-1.7B Base, REVO at 50 training steps surpasses standard OPD at 200 steps, measured by mean Avg@16 over AIME24, AIME25, and AMC23.

- We develop a rollout-reuse framework that separates prefix mismatch from current-token mismatch. Specifically, we propose one-step resampling, which draws fresh tokens to mitigate the distributional shift, while stabilized prefix weighting supports repeated updates on stored trajectories.

- We introduce bounded token reweighting based on the local relative-entropy variance, which describes the variance of the log-probability ratio between the student and teacher models. A Fisher-geometric analysis connects this statistic to local natural-gradient progress, and finite-sample diagnostics support its use for prioritizing informative token-level updates during rollout reuse.

- Empirically, REVO trained with only 50 rollout iterations matches or exceeds OPD baselines trained for 200 iterations across three student–teacher scales and a range of in-domain and cross-domain reasoning benchmarks; ablations further isolate the contribution of each component.

2 Related Work

2.1 Knowledge Distillation and On-Policy Distillation

Classical knowledge distillation trains a student on fixed data using supervision from a stronger teacher (Hinton et al., 2015; Agarwal et al., 2024). On-policy distillation instead queries the teacher on student-generated prefixes, providing dense supervision on states visited by the current student policy (Agarwal et al., 2024; Team et al., 2026; DeepSeek-AI, 2026; Yang et al., 2025). Recent OPD methods commonly optimize token-level reverse KL (Lu & Lab, 2025; Li et al., 2026a). MiniLLM (Gu et al., 2024) studies the corresponding sequence-level objective, while other methods use forward KL (Zhao et al., 2026; Jin et al., 2026). AsyncOPD (Kang et al., 2026) studies policy staleness in asynchronous rollout and learner pipelines, with a focus on systems throughput. Recently, using the model itself as a teacher with privileged information has also emerged as a promising direction (Zhao et al., 2026; Yang et al., 2026a;b). However, performing multiple off-policy learner updates on each rollout batch has received less attention in OPD.

2.2 Off-Policy Optimization in RLVR

LLM reinforcement learning is predominantly on-policy, with each round of optimization using fresh rollouts from the current or a recent policy. This keeps the training distribution well aligned but makes generation a major cost. Recent work shows that stale trajectories can be reused when the resulting distribution shift is properly controlled (Arnal et al., 2026; Zheng et al., 2026). Token-level importance ratios correct only the sampled action at a fixed prefix, while exact trajectory-level ratios account for the full distribution shift but can suffer from rapidly growing variance with sequence length. This has motivated stabilized alternatives such as sequence-level geometric ratios (Zheng et al., 2025) and cumulative token importance sampling (Zhang et al., 2026). These methods are developed for RLVR, where supervision is tied to the reward of the stored trajectory and optimization remains anchored to its sampled actions. Distillation offers additional flexibility because the teacher can score arbitrary actions at a stored prefix. We exploit this property by resampling the current token from the current student and retaining only a prefix-level correction.

2.3 Learning-Signal Prioritization

A broad line of reinforcement learning research allocates training effort according to the estimated learning value of individual samples or tasks. Prioritized experience replay uses temporal-difference error as a proxy for learning progress (Schaul et al., 2016; Lahire et al., 2022), while curriculum and level replay methods prioritize tasks based on estimated learning potential or progress (Jiang et al., 2021; Portelas et al., 2020).

Related ideas have recently appeared in RLVR for language models. DAPO (Yu et al., 2025) filters prompt groups with zero reward variance, while Dr. GRPO (Liu et al., 2025) removes within-group reward standard deviation normalization. LILO (Foster et al., 2025) more explicitly characterizes within-prompt reward variance as *learnability*. For binary rewards, this quantity reduces to $p(1-p)$ and becomes small for prompts that are consistently solved or consistently failed.

We consider learning-signal prioritization at the token level under reverse-KL distillation. The remaining distillation signal can vary substantially across token positions, especially when stored prefixes are reused for multiple learner updates. We use relative-entropy variance, which we refer to as ***RKL variance***, to quantify this token-level signal and prioritize positions during rollout reuse. We develop its optimization interpretation in Section 4.2.

3 Preliminaries

On-policy distillation. Let $c_t = (x, y_{<t})$ denote the context at position t, and let $P_\theta(\cdot \mid c_t)$ and $P_T(\cdot \mid c_t)$ denote the student and teacher next-token distributions. Token-level on-policy distillation minimizes the reverse KL divergence $\mathcal{L}_{\mathrm{OPD}}(c_t) = D_{\mathrm{KL}}(P_\theta(\cdot \mid c_t) \| P_T(\cdot \mid c_t))$. Define the token-level reverse-KL signal as $A_\theta(c_t, a) = \log \frac{P_\theta(a|c_t)}{P_T(a|c_t)}$. When the context is clear, we abbreviate

$A_t(a) := A_\theta(c_t, a)$. The local gradient is

$$\nabla_\theta \mathcal{L}_{\mathrm{OPD}}(c_t) = \mathbb{E}_{a\sim P_\theta(\cdot|c_t)} \left[A_\theta(c_t, a) \nabla_\theta \log P_\theta(a \mid c_t) \right],$$

where the additional score-function term vanishes in expectation.

Computing this expectation over the full vocabulary is expensive for large language models. Sampled-token OPD (Lu & Lab, 2025) instead uses the rollout token $a_t = y_t \sim P_\theta(\cdot \mid c_t)$ at each visited context, yielding an unbiased gradient estimator. We use this formulation throughout our experiments. Top-k approximations (Li et al., 2026a) restrict the expectation to high-probability tokens and introduce truncation bias. In our experiments, Top-k OPD does not outperform sampled-token OPD (Figure 2), consistent with recent empirical findings (Team et al., 2026).

Off-policy reverse-KL distillation. Under rollout reuse, trajectories are generated by a rollout policy P_o, while optimization is performed under the current student P_θ. Let $d_o(c_t)$ and $d_\theta(c_t)$ denote the prefix distributions induced by the two policies. Let g_t^{on} denote the expected sampled-token OPD gradient under $c_t \sim d_\theta$ and $a_t \sim P_\theta(\cdot \mid c_t)$.

To express this gradient using trajectories generated by P_o, define the cumulative importance ratio

$$W_t = \frac{d_\theta(c_t) P_\theta(a_t \mid c_t)}{d_o(c_t) P_o(a_t \mid c_t)} = \prod_{j\le t} \frac{P_\theta(a_j \mid c_j)}{P_o(a_j \mid c_j)}, \tag{1}$$

where the prompt distribution is shared by the two policies. A change of measure gives

$$g_t^{\mathrm{on}} = \mathbb{E}_{c_t\sim d_o,\, a_t\sim P_o(\cdot|c_t)} \left[W_t A_\theta(c_t, a_t) \nabla_\theta \log P_\theta(a_t \mid c_t) \right].$$

The exact correction W_t multiplies likelihood ratios along the trajectory, which can produce highly variable weights for long sequences. We address this instability in the next section.

4 Method

4.1 Stabilizing Off-Policy RKL Distillation

Directly applying the cumulative importance ratio in Equation (1) is poorly suited to long LLM trajectories. Writing $W_t = W_{<t} r_t$, where $W_{<t} = d_\theta(c_t)/d_o(c_t)$ corrects the stored prefix distribution and $r_t = P_\theta(a_t \mid c_t)/P_o(a_t \mid c_t)$ corrects the current-token distribution, reveals two distinct sources of mismatch. We treat these two terms differently. The current token can be resampled from the latest student policy at a stored prefix, while refreshing the prefix itself would require generating a new trajectory. We therefore remove the current-token ratio through resampling and replace the cumulative prefix ratio with a stabilized surrogate.

One-step resampling from the current policy. Conditioned on a stored prefix c_t, the current-token importance ratio can be removed exactly by a change of measure:

$$\mathbb{E}_{a_t\sim P_o(\cdot|c_t)} \left[r_t A_t(a_t) \nabla_\theta \log P_\theta(a_t \mid c_t) \right] = \mathbb{E}_{a_t\sim P_\theta(\cdot|c_t)} \left[A_t(a_t) \nabla_\theta \log P_\theta(a_t \mid c_t) \right]. \tag{2}$$

We therefore draw $a_t^{(1)}, \ldots, a_t^{(K)} \overset{\text{i.i.d.}}{\sim} P_\theta(\cdot \mid c_t)$ at every learner update and use their average to obtain an unbiased estimate of the conditional reverse-KL gradient. We write $A_t^{(k)} := A_t(a_t^{(k)})$ for the reverse-KL signal of the k-th candidate. This requires no additional autoregressive rollout because the stored prefix remains unchanged. The teacher can score all resampled tokens at the same prefix.

Stabilizing the prefix correction. The prefix mismatch cannot be removed in the same way, since sampling a new prefix from the current student would require regenerating the preceding trajectory. The exact prefix ratio accumulates token likelihood ratios along the stored prefix and can become highly variable for long contexts. We therefore replace it with a geometrically normalized correction, motivated by the stabilization used in Zheng et al. (2025):

$$G_{<t} = \min \left\{ \exp\left(\tfrac{1}{t-1} \textstyle\sum_{i<t} \log \frac{P_\theta(a_i|c_i)}{P_o(a_i|c_i)} \right), C \right\}, \tag{3}$$

with $G_{<1} = 1$, where C bounds the correction to keep updates stable. Below the cap, $G_{<t} = W_{<t}^{1/(t-1)}$ reflects the average policy shift along the stored prefix instead of growing multiplicatively with prefix length, giving a stabilized surrogate rather than an exact importance correction. Unlike GSPO, which constructs a ratio over the full generated response, our weight is prefix-dependent and only uses the policy mismatch accumulated before position t.

Stabilized training objective. Combining one-step resampling with the stabilized prefix correction gives the practical surrogate used by REVO:

$$\widehat{\mathcal{L}}_t^{K,G} = \frac{1}{K} \sum_{k=1}^{K} \mathrm{sg}\Big[G_{<t} A_t^{(k)}\Big] \log P_\theta(a_t^{(k)} \mid c_t), \tag{4}$$

where $a_t^{(k)} \sim P_\theta(\cdot \mid c_t)$. The correction weight and reverse-KL signal are treated as constants in the stop-gradient surrogate. This objective removes direct importance weighting of the current token while retaining a stable correction for the mismatch of the reused prefix.

4.2 Variance-Guided Token Prioritization

In Section 4.1, the current token is resampled at every learner update, but the stored prefixes remain fixed for all N updates. Refreshing them would require new rollouts, which is the cost reuse is intended to avoid. The prefix weight $G_{<t}$ stabilizes updates under policy shift, but it does not indicate how much the student can still learn from the teacher at each prefix. A prefix can be fully on-policy yet provide little learning signal if the student already matches the teacher there. Given the uneven learning signals across token positions (Jin et al., 2026; Li et al., 2026b), repeatedly revisiting the same prefixes can spend additional computation on positions with little left to contribute. We therefore complement off-policy stabilization with adaptive prioritization of the remaining token-level learning signal.

RKL variance and the usefulness of reused supervision. For a stored prefix c_t, let $p_t(a) = P_\theta(a \mid c_t)$ and $q_t(a) = P_T(a \mid c_t)$, so that $A_t(a) = \log \frac{p_t(a)}{q_t(a)}$. We measure its variation across candidate next tokens using the *local relative-entropy variance*, $E_t = \mathrm{Var}_{a \sim p_t}[A_t(a)]$, which we call the *RKL variance*. This statistic depends on both the teacher and the current student and can be estimated from the same tokens used for one-step resampling.

We first examine whether RKL variance distinguishes token positions that contribute useful supervision during rollout reuse. Figure 3 compares training with updates restricted to positions above or below a fixed RKL-variance threshold. The high-variance subset yields sustained performance gains, whereas the low-variance subset provides little improvement. This separation supports using RKL variance to prioritize positions during rollout reuse. Our practical rule retains both groups with positive weights. We next examine the reliability of the corresponding gradient estimates.

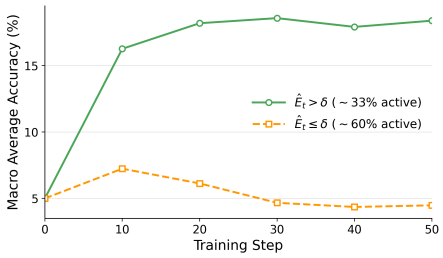

Figure 3: High- versus low-RKL-variance updates under REVO for Qwen3-1.7B Base→4B ($\delta = 0.005$). Accuracy is mean Avg@16 on AIME24, AIME25, and AMC23.

RKL variance as a proxy for gradient reliability. Let $z_t = z_\theta(c_t)$ denote the student logits at c_t, let $u_t = \nabla_{z_t} D_{\mathrm{KL}}(p_t \| q_t)$ be the exact logit gradient, and let $\widehat{u}_t$ be its unbiased K-sample estimate from one-step resampling. We define the gradient signal-to-noise ratio (SNR) as $\Gamma_t = \|u_t\|_2^2 / \sigma_t^2$, where $\sigma_t^2 = \mathbb{E}\|\widehat{u}_t - u_t\|_2^2$. The two variances measure different quantities: E_t captures log-ratio variation across actions, whereas σ_t^2 measures sampling noise in the gradient estimator.

Figure 4 shows a strong positive rank correlation of 0.93 between exact E_t and the SNR of the $K = 16$ resampling estimator across 1280 stored prefixes. Higher RKL-variance positions are associated with more reliable logit-gradient estimates in this diagnostic, consistent with their greater training utility in Figure 3. These quantities are computed offline from full-vocabulary student and

teacher distributions (Section B.1). This association motivates using RKL variance as a proxy for relative gradient reliability without computing full-vocabulary gradient moments during training.

Variance-aware weighted reverse-KL minimization. Gradient reliability provides a principle for allocating update weights. The following proposition characterizes the optimal deterministic scaling of the logit-gradient estimator $\widehat{u}_t$ under a smoothness-based bound.

Proposition 4.1 (SNR-guided local update scaling)**.** *Fix a stored prefix c_t and consider the local reverse-KL objective as a function of the student logits,*

$$\mathcal{L}_t(z_t) = D_{\mathrm{KL}}(\mathrm{softmax}(z_t)\|q_t).$$

Let $\beta > 0$ be a common smoothness upper bound over the stored prefixes under consideration, and suppose $\mathcal{L}_t$ is β-smooth. Let $u_t = \nabla_{z_t}\mathcal{L}_t(z_t)$ and let $\widehat{u}_t$ be an unbiased estimator with $\sigma_t^2 = \mathbb{E}\|\widehat{u}_t - u_t\|_2^2 \in (0,\infty)$. Define

$$\Gamma_t = \frac{\|u_t\|_2^2}{\sigma_t^2}.$$

For any step size $\eta > 0$ and deterministic weight $w \geq 0$,

$$\mathbb{E}\big[\mathcal{L}_t(z_t - \eta w \widehat{u}_t)\big] \leq \mathcal{L}_t(z_t) - \eta w\|u_t\|_2^2 + \frac{\beta\eta^2 w^2}{2}\big(\|u_t\|_2^2 + \sigma_t^2\big).$$

The right-hand side is uniquely minimized over $w \geq 0$ at

$$w_t^\star = \frac{1}{\beta\eta}\frac{\Gamma_t}{1+\Gamma_t}. \tag{5}$$

For fixed β and η, $w_t^\star$ increases monotonically with Γ_t and approaches $1/(\beta\eta)$ as $\Gamma_t \to \infty$.

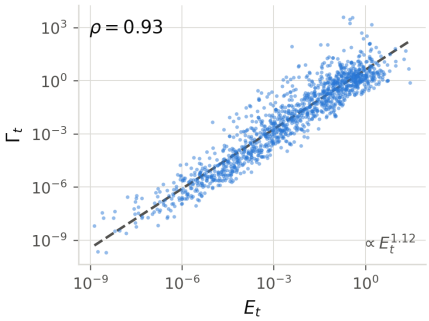

Figure 4: RKL variance and gradient reliability across 1,280 stored prefixes. Exact E_t is strongly rank-correlated with the logit-space gradient SNR of the $K = 16$ resampling estimator, with Spearman's $\rho = 0.93$.

The proof is given in Section A.2. Proposition 4.1 provides an oracle local criterion under which the preferred scale increases with gradient reliability and saturates at high SNR. Together with the strong rank association between E_t and Γ_t, this motivates using RKL variance to construct bounded token priorities.

During training, we estimate E_t from the K resampled tokens using the unbiased sample variance,

$$\widehat{E}_t^K = \tfrac{1}{K-1}\textstyle\sum_{k=1}^K \big(A_t^{(k)} - \tfrac{1}{K}\sum_{j=1}^K A_t^{(j)}\big)^2. \tag{6}$$

These estimates span several orders of magnitude and exhibit a long right tail; finite sampling can also produce zero estimated variance at positions with nonzero underlying signal (Section B.2). We therefore use a threshold $\delta \geq 0$ to recover a coarse priority ordering with bounded weights for both groups:

$$\widehat{w}_t = \left|\tau - \mathbb{1}\left\{\widehat{E}_t^K \leq \delta\right\}\right|, \qquad \tau \in [1/2, 1). \tag{7}$$

Positions above the threshold receive weight τ, while the remaining positions retain weight $1-\tau$. Normalizing these weights to have unit mean over all valid token positions in the stored batch gives w_t. For $\tau > 1/2$, this rule emphasizes the higher-variance group while retaining supervision at every valid position. When $\tau = 1/2$, it recovers uniform weighting.

Combining these priorities with Equation (4) yields the adaptive weighted surrogate

$$\widehat{\mathcal{L}}_t^{K,G,w} = \frac{1}{K}\sum_{k=1}^K \mathrm{sg}\Big[G_{<t}\, w_t\, A_t^{(k)}\Big] \log P_\theta(a_t^{(k)} \mid c_t). \tag{8}$$

We average this surrogate over valid positions and treat the weights and reverse-KL signals as constants during differentiation. Samples and priorities are refreshed at every learner update, allowing the weighting to adapt to the evolving student without new trajectories or additional model evaluations for variance estimation. The complete procedure is given in Section B.4. We further test continuous variants of this weighting, using $\sqrt{\widehat{E}_t^K}$ and a saturating transformation of $\widehat{E}_t^K$ in place of the two-level rule. Both underperform the two-level rule (Table 3), supporting this design choice.

Interpretation through local optimization dynamics. We finally examine what RKL variance measures in local optimization. At a fixed prefix, the score function has zero mean, so only the centered log-ratio $A_t - \mathbb{E}_{p_t}[A_t]$ contributes to the exact gradient. The following proposition relates this variation to progress under conditional Fisher geometry.

Proposition 4.2 (Fisher-geometric interpretation of RKL variance)**.** *Fix a prefix c_t, and let $g_t = \nabla_\theta D_{\mathrm{KL}}(p_t \| q_t)$. Define $\psi_t(a) = \nabla_\theta \log p_t(a)$, the conditional Fisher matrix $F_t = \mathbb{E}_{a\sim p_t}[\psi_t(a)\psi_t(a)^\top]$, and the natural gradient $n_t = F_t^\dagger g_t$, where $\dagger$ denotes the Moore–Penrose pseudoinverse. If the model can locally realize the centered log-ratio correction at c_t, then*

$$\langle g_t, n_t \rangle = g_t^\top F_t^\dagger g_t = E_t.$$

The precise realizability condition and proof are given in Section A.1.

Under this condition, E_t gives the first-order decrease per unit step along the negative natural-gradient direction. Thus, E_t measures the local optimization signal remaining at a prefix, while Figure 4 relates it to the reliability of the finite-sample gradient estimate. Together, these properties motivate refreshing variance-guided priorities throughout rollout reuse.

5 Experiments

5.1 Experimental Setup

Models and benchmarks. We conduct experiments on the Qwen3 model family (Yang et al., 2025). Specifically, we initialize Qwen3-0.6B-Base, Qwen3-1.7B-Base, and Qwen3-4B-Base as student models, and use Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively, as teachers. We use DAPO-Math-17K (Yu et al., 2025) as the training dataset.

To evaluate the performance of REVO, we primarily consider mathematical reasoning benchmarks, including AIME24, AIME25, AMC23, MATH-500, OlympiadBench, and Minerva (Zhang & Math-AI, 2024; 2025; Mathematical Association of America, 2025; Lewkowycz et al., 2022; He et al., 2024). For smaller benchmarks, namely AIME24, AIME25, and AMC23, we report Avg@16, while for the larger benchmarks we report Avg@4.

We further evaluate all methods on benchmarks beyond mathematical reasoning to assess cross-domain generalization, including GPQA@4, HumanEval@4, and MMLU-Redux@1 (Rein et al., 2023; Chen et al., 2021; Gema et al., 2024).

Baselines. We mainly compare REVO against sampled-token OPD (Lu & Lab, 2025) (implemented following Li et al. (2026a)), EOPD (Jin et al., 2026), and FiRe (Li et al., 2026b). We train REVO for 50 rollout steps, performing 10 learner updates per step. We compare against the baselines under both the same rollout budget (50 steps) and a larger training budget of 200 steps.

Implementation details. Each rollout step uses 8 prompts with 4 responses per prompt. We preserve the original ordering of the training data without shuffling. For the geometric-mean prefix correction, we set the upper clipping bound to $C = 4$, and use $K = 16$ candidates for one-step resampling. For RKL-variance-guided reweighting, we set the threshold to $\delta = 0.005$ and $\tau = 0.75$.

We disable the teacher's thinking mode and use a maximum sequence length of 8192 tokens during training. For evaluation, we use a maximum generation budget of 8192 tokens, with temperature 0.7 and top-p sampling with $p = 0.95$. Detailed configurations are provided in Section C.1, and additional results and ablations in Sections B and C.

5.2 Main Results

We report the main results in Tables 1 and 2. REVO consistently outperforms other OPD variants under the same rollout budget and surpasses baselines trained with a $4\times$ larger budget. This pattern extends beyond Qwen3, with REVO at 50 rollout iterations outperforming 200-iteration baselines on Llama 3.2 and Gemma 3 (Section B.3). These results demonstrate improved rollout efficiency. Wall-clock comparisons and larger-batch OPD results are provided in Sections C.5 and C.6.

Table 1: Results across three student–teacher scales. AIME24, AIME25, and AMC23 use Avg@16; MATH-500, OlympiadBench, and Minerva use Avg@4. Mean is the average across the six benchmarks. Best results within each scale are bold; blue shading highlights REVO. Green/red subscripts indicate gains/losses in percentage points relative to OPD at Step 50 within the same scale.

		Avg@16			Avg@4			
Method	**Step**	**AIME24**	**AIME25**	**AMC23**	**MATH-500**	**OlympiadBench**	**Minerva**	**Mean**
Student: 0.6B Base	**Teacher: 1.7B**							
Base	0	0.0	0.0	1.4	3.3	1.7	2.3	1.5
OPD	50	0.8	1.0	18.4	45.2	14.8	11.0	15.2
OPD	200	**2.1**	1.0	23.3	48.4	17.9	10.0	17.1
EOPD	50	1.5	0.6	18.8	45.0	13.3	10.7	15.0
EOPD	200	1.9	0.6	24.7	47.4	17.6	11.8	17.3
FiRe	50	1.2	0.6	22.8	45.1	15.0	10.7	15.9
FiRe	200	**2.1**	**1.2**	26.2	47.6	17.5	10.7	17.6
REVO	50	**2.1**$_{+1.3}$	0.8$_{-0.2}$	**29.8**$_{+11.4}$	**49.0**$_{+3.8}$	**18.2**$_{+3.4}$	**12.5**$_{+1.5}$	**18.7**$_{+3.5}$
Student: 1.7B Base	**Teacher: 4B**							
Base	0	0.4	1.3	13.8	18.8	5.5	6.7	7.8
OPD	50	6.5	4.2	34.7	64.3	26.7	17.4	25.6
OPD	200	8.3	6.5	35.6	66.9	28.8	17.4	27.3
EOPD	50	6.9	3.5	32.7	59.3	24.1	16.0	23.8
EOPD	200	9.2	5.2	36.7	67.3	29.2	15.3	27.2
FiRe	50	5.4	4.1	36.2	63.1	26.5	18.5	25.6
FiRe	200	7.2	5.6	35.0	66.6	28.4	16.9	26.6
OPD(U10)	50	7.5	4.7	32.8	62.6	27.1	17.3	25.3
REVO	50	**10.0**$_{+3.5}$	**7.3**$_{+3.1}$	**41.6**$_{+6.9}$	**69.0**$_{+4.7}$	**29.7**$_{+3.0}$	**19.9**$_{+2.5}$	**29.6**$_{+4.0}$
Student: 4B Base	**Teacher: 8B**							
Base	0	10.4	7.7	35.3	41.0	23.7	6.7	20.8
OPD	50	14.8	16.5	55.9	78.2	40.1	26.7	38.7
OPD	200	16.3	17.3	57.8	78.8	41.5	22.9	39.1
EOPD	50	16.3	14.2	53.9	78.5	41.1	24.9	38.2
EOPD	200	16.7	16.0	56.1	79.5	43.7	23.2	39.2
FiRe	50	14.7	13.5	51.7	77.4	41.0	26.4	37.5
FiRe	200	**18.7**	15.8	56.5	78.9	40.3	22.2	38.7
REVO	50	18.1$_{+3.3}$	**17.5**$_{+1.0}$	**58.0**$_{+2.1}$	**80.4**$_{+2.2}$	**45.1**$_{+5.0}$	**29.0**$_{+2.3}$	**41.4**$_{+2.7}$

Table 2: Cross-domain evaluation across three student–teacher scales. All experiments distill a larger post-trained teacher into a smaller Qwen3 Base student. REVO is evaluated at Step 50, while OPD, EOPD, and FiRe are evaluated at Step 200. Mean is the unweighted average across the three benchmarks. Best results within each scale are bold; blue shading highlights REVO.

	Student: 0.6B Base		**Teacher: 1.7B**		**Student: 1.7B Base**		**Teacher: 4B**		**Student: 4B Base**		**Teacher: 8B**	
Benchmark	**REVO**	OPD	EOPD	FiRe	**REVO**	OPD	EOPD	FiRe	**REVO**	OPD	EOPD	FiRe
GPQA@4	**20.2**	13.8	18.1	13.3	**27.3**	17.5	20.8	16.8	**40.0**	38.5	37.7	38.8
HumanEval@4	**32.6**	30.8	29.7	31.6	**54.3**	51.2	52.3	49.2	**77.9**	75.8	77.0	75.2
MMLU-Redux@1	**48.0**	46.9	46.9	45.0	**65.2**	63.2	63.8	62.1	79.6	79.0	**80.1**	79.5
Mean	**33.6**	30.5	31.6	29.9	**48.9**	44.0	45.6	42.7	**65.8**	64.4	64.9	64.5

We further compare REVO with a multi-update baseline, OPD(U10), which directly applies the standard OPD objective for 10 learner updates per rollout step. This matches both the rollout budget and number of learner updates. On the 1.7B student with 50 rollout steps, OPD(U10) performs substantially worse than REVO and provides no clear improvement over standard OPD under the same rollout budget. This shows that the gains of REVO do not come simply from more learner updates, highlighting the importance of accounting for off-policy reuse.

5.3 Ablation Study

We further isolate the contribution of each component in Table 3. Repeating learner updates on stored rollouts without off-policy correction provides essentially no improvement over standard OPD, with Avg@16 remaining at 15.0 compared with 15.1. We then compare two al-

ternative corrections for the current token. PPO-style clipped importance sampling improves performance to 16.7, while one-step resampling performs better at 17.8. Starting from the one-step resampling variant, adding prefix correction further improves performance to 18.2.

Table 3: Ablation study using a Qwen3-1.7B Base student distilled from a Qwen3-4B teacher. Results are reported at Step 50 as Avg@16 over AIME24, AIME25, and AMC23. PPO-style clipped IS and one-step resampling are alternative current-token corrections. Prefix correction and RKL-variance weighting are added to the one-step resampling variant.

Method	**Avg@16**
OPD	15.1
OPD(U10)	15.0
OPD(U10) + PPO-Style Clipped IS	16.7
OPD(U10) + One-Step Resampling	17.8
+ Prefix Correction	18.2
RKL-variance reweighting	
+ $\sqrt{\widehat{E}_t^K}$ weighting	18.7
+ $\widehat{E}_t^K/(\widehat{E}_t^K + 0.25)$	18.6
+ Two-Level Weighting (REVO)	**19.6**

On top of one-step resampling and prefix correction, we compare three RKL-variance weighting rules. The continuous $\sqrt{\widehat{E}_t^K}$ and saturating $\widehat{E}_t^K/(\widehat{E}_t^K + c)$ rules with $c = 0.25$ achieve 18.7 and 18.6, respectively, both outperforming uniform weighting. The bounded two-level rule performs best at 19.6. This suggests that the relative ordering induced by RKL variance is more useful than its raw magnitude, supporting its use as a relative priority signal rather than directly determining the update magnitude.

Additional analysis of the empirical RKL-variance distribution is provided in Section B.2. Implementation details for the alternative weighting rules are given in Section C.1. Further ablations on hyperparameter sensitivity and alternative prioritization signals are provided in Sections C.3 and C.4, respectively, to evaluate the robustness of the weighting rule and the choice of prioritization signal.

Effect of the number of updates. Our default configuration applies 10 learner updates to each rollout batch. We further increase the reuse rate to 50 updates per batch (U50) to study whether more aggressive off-policy optimization can further improve rollout efficiency. As shown in Figure 5, increasing the number of updates substantially accelerates early learning. U50 reaches strong performance within only a few rollout iterations, showing that REVO remains effective under more aggressive rollout reuse.

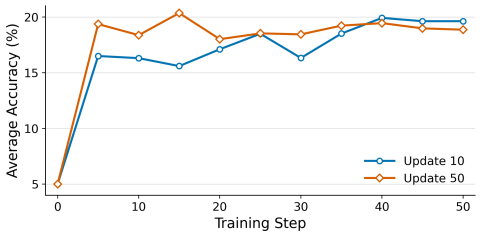

Figure 5: Effect of the number of learner updates per rollout batch. Increasing the reuse rate accelerates early learning, while both configurations eventually reach similar performance.

The two configurations eventually converge to a similar performance level, and U50 provides no improvement in final accuracy despite requiring $5\times$ more learner updates per rollout batch. Additional reuse therefore mainly trades learner computation for fewer rollouts. We use U10 as the default setting for a better balance between rollout efficiency and optimization cost, while U50 can reduce the number of required rollout iterations when generation is the primary bottleneck.

6 Conclusion

In this work, we propose REVO, a practical framework for off-policy reverse-KL distillation with repeated rollout reuse. Starting from the standard OPD objective, we derive the corresponding off-policy formulation and introduce stabilized prefix-level correction together with current-policy one-step resampling to make multi-update training practical. We further introduce *RKL variance* as a token-level measure of remaining reverse-KL learning signal and use it to allocate optimization effort more effectively across reused prefixes. Extensive experiments across multiple student and teacher scales demonstrate that REVO enables substantially more rollout-efficient distillation, matching or outperforming OPD baselines trained with $4\times$ more rollout iterations. Overall, our results suggest that effective rollout reuse benefits from both off-policy stabilization and adaptive allocation of optimization effort across token positions.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, guidance on figure organization and presentation, assistance with code implementation and modification under author-provided instructions, and assistance with checking mathematical derivations and proofs. All AI-assisted outputs were reviewed and verified by the authors. The authors take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies the distillation of student language models from stronger teacher models. All experiments use publicly available open-source models, datasets, and evaluation benchmarks. The study does not involve human subjects, private user data, or deployment in high-stakes settings. We do not identify ethical concerns specific to this study beyond those generally associated with training and evaluating language models.

REPRODUCIBILITY STATEMENT

We provide the models, benchmarks, and baselines used in our experiments in Section 5.1. Implementation details and hyperparameters are reported in Section C.1, and the complete training procedure is summarized in Algorithm 1. We also include anonymized code in the supplementary material to facilitate reproduction of our results.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Charles Arnal, Vivien Cabannes, Taco Cohen, Julia Kempe, and Remi Munos. Efficient rl training for llms with experience replay, 2026. URL https://arxiv.org/abs/2604.08706.

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

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Thomas Foster, Anya Sims, Johannes Forkel, and Jakob Foerster. Lilo: Learning to reason at the frontier of learnability. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 48941–48974. Curran Associates, Inc., 2025. doi: 10.52202/085713-1634. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/45d5823950cab9a88e10c6e8dbff351c-Paper-Conference.pdf.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are we done with mmlu?, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jiang21b.html.

Woogyeol Jin, Taywon Min, Yongjin Yang, Dennis Wei, Yi Zhou, Swanand Ravindra Kadhe, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models, 2026. URL https://arxiv.org/abs/2603.07079.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjun Kang, Sanghyun Park, Donghoon Kim, Minjae Lee, Minseo Kim, Rishabh Tiwari, Yuchen Zeng, Hyung Il Koo, and Kangwook Lee. Asyncopd: How stale can on-policy distillation be?, 2026. URL https://arxiv.org/abs/2606.24143.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A contrastive approach boosts the distillation of LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=rc65N9xIrY.

Thibault Lahire, Matthieu Geist, and Emmanuel Rachelson. Large batch experience replay. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 11790–11813. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/lahire22a.html.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857. Curran Associates, Inc., 2022. doi: 10.52202/068431-0278. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe, 2026a. URL https://arxiv.org/abs/2604.13016.

Yuying Li, Leqi Zheng, Yongzi Yu, Wenrui Zhou, Xuchang Zhong, Xing Hu, Jing Jin, Hangjie Yuan, and Tao Feng. Filter, then reweight: Rethinking optimization granularity in on-policy distillation, 2026b. URL https://arxiv.org/abs/2606.02684.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Mathematical Association of America. American mathematics competitions, 2025. Accessed: 2025-09-13.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 835–853. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/portelas20a.html.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016. URL https://arxiv.org/abs/1511.05952.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai, Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin,

Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr, 2026a. URL https://arxiv.org/abs/2604.03128.

Yuxiao Yang, Xiaoyun Wang, and Weitong Zhang. Ogls-sd: On-policy self-distillation with outcome-guided logit steering for llm reasoning, 2026b. URL https://arxiv.org/abs/2605.12400.

Yuxiao Yang, Tianrun Yu, Shangzhe Li, Kaixiang Zhao, Xuchao Zhang, Chetan Bansal, Huaxiu Yao, Taylor W. Killian, and Weitong Zhang. When eos tokens disagree: Understanding length inflation in on-policy distillation, 2026c. URL https://arxiv.org/abs/2609.20511.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

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, 2026. URL https://arxiv.org/abs/2605.07331.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Haizhong Zheng, Jiawei Zhao, and Beidi Chen. Prosperity before collapse: How far can off-policy rl reach with stale data on llms? In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 82657–82679, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8590b946e4f0c2b1262cb5b047924653-Paper-Conference.pdf.

Zhou Ziheng, Jiaqi Li, Huacong Tang, Ying Nian Wu, and Demetri Terzopoulos. Less is more: Early stopping rollout for on-policy distillation, 2026. URL https://arxiv.org/abs/2605.27028.

A Mathematical Proofs

A.1 Proof of the OPD–NPG Gradient Identity

We first introduce the notation and establish the lemmas needed for Proposition 4.2. In what follows, we write s for a fixed prefix c_t and fix a parameter value $\theta \in \mathbb{R}^d$. Let $z_\theta(s) \in \mathbb{R}^V$ denote the student logits over a finite vocabulary $\mathcal{V}$, with $|\mathcal{V}| = V$, and let

$$p_s(a) = P_\theta(a \mid s), \qquad q_s(a) = P_T(a \mid s), \qquad p_s = \mathrm{softmax}(z_\theta(s)).$$

We assume that the teacher distribution q_s is fixed, both distributions have full support, and the logits are twice continuously differentiable in a neighborhood of θ.

The local OPD objective is

$$L_s(\theta) = D_{\mathrm{KL}}(p_s \| q_s) = \sum_{a \in \mathcal{V}} p_s(a) \log \frac{p_s(a)}{q_s(a)}. \tag{9}$$

Define the log-ratio signal, its mean, and its centered version as

$$A_s(a) = \log p_s(a) - \log q_s(a), \qquad \bar{A}_s = \mathbb{E}_{a \sim p_s}[A_s(a)] = L_s(\theta), \qquad X_s(a) = A_s(a) - \bar{A}_s.$$

We also use $A_s, X_s \in \mathbb{R}^V$ to denote the vectors containing these action-wise values. By construction,

$$p_s^\top X_s = \sum_{a \in \mathcal{V}} p_s(a) X_s(a) = 0. \tag{10}$$

The local relative-entropy variance is

$$E_s = \mathrm{Var}_{a \sim p_s}[A_s(a)] = \sum_{a \in \mathcal{V}} p_s(a) X_s(a)^2. \tag{11}$$

Define the logit Jacobian and categorical Fisher matrix by

$$J_s = \frac{\partial z_\theta(s)}{\partial \theta} \in \mathbb{R}^{V \times d}, \qquad G_s = \mathrm{Diag}(p_s) - p_s p_s^\top \in \mathbb{R}^{V \times V}. \tag{12}$$

Define the score vector

$$\psi_s(a) := \nabla_\theta \log p_s(a) \in \mathbb{R}^d, \tag{13}$$

and the corresponding conditional Fisher matrix

$$F_s := \mathbb{E}_{a \sim p_s}\left[\psi_s(a) \psi_s(a)^\top\right] \in \mathbb{R}^{d \times d}. \tag{14}$$

All expectations in this subsection are taken at the fixed prefix s. For any matrix M, $M^\dagger$ denotes its Moore–Penrose pseudoinverse.

Lemma 1 (Local OPD gradient)**.** *The parameter gradient of the local OPD objective satisfies*

$$g_s := \nabla_\theta L_s(\theta) = J_s^\top G_s X_s. \tag{15}$$

Proof. For the softmax distribution,

$$\frac{\partial \log p_s(a)}{\partial z_s(j)} = \mathbf{1}\{a = j\} - p_s(j).$$

Therefore, by the chain rule,

$$\psi_s(a) = J_s^\top (\mathbf{e}_a - p_s), \tag{16}$$

where $\mathbf{e}_a \in \mathbb{R}^V$ is the one-hot vector corresponding to action a.

The score has zero expectation:

$$\begin{aligned} \sum_a p_s(a) \psi_s(a) &= \sum_a p_s(a) \nabla_\theta \log p_s(a) \\ &= \sum_a \nabla_\theta p_s(a) \\ &= \nabla_\theta \sum_a p_s(a) = 0. \end{aligned} \tag{17}$$

Differentiating $L_s(\theta) = \sum_a p_s(a)A_s(a)$ gives

$$g_s = \sum_a A_s(a)\nabla_\theta p_s(a) + \sum_a p_s(a)\nabla_\theta A_s(a). \tag{18}$$

Because the teacher is fixed, $\nabla_\theta A_s(a) = \psi_s(a)$, so the second term vanishes by the zero-mean score identity. Using $\nabla_\theta p_s(a) = p_s(a)\psi_s(a)$ in the first term gives

$$\begin{aligned} g_s &= \sum_a p_s(a)A_s(a)\psi_s(a) \\ &= \sum_a p_s(a)\big(X_s(a) + \bar{A}_s\big)\psi_s(a) \\ &= \sum_a p_s(a)X_s(a)\psi_s(a), \end{aligned} \tag{19}$$

where the constant term vanishes again by the zero-mean score identity.

Substituting $\psi_s(a) = J_s^\top(\mathbf{e}_a - p_s)$, we obtain

$$\begin{aligned} g_s &= \sum_a p_s(a)X_s(a)J_s^\top(\mathbf{e}_a - p_s) \\ &= J_s^\top\left[\sum_a p_s(a)X_s(a)\mathbf{e}_a - p_s\sum_a p_s(a)X_s(a)\right] \\ &= J_s^\top\left[\mathrm{Diag}(p_s)X_s - p_s(p_s^\top X_s)\right] \\ &= J_s^\top\left[\mathrm{Diag}(p_s) - p_s p_s^\top\right]X_s \\ &= J_s^\top G_s X_s. \end{aligned} \tag{20}$$

□

Lemma 2 (Parameter-space natural gradient)**.** *The conditional parameter Fisher satisfies*

$$F_s = J_s^\top G_s J_s. \tag{21}$$

The parameter-space natural gradient of L_s at the fixed prefix is

$$n_s = F_s^\dagger g_s = (J_s^\top G_s J_s)^\dagger J_s^\top G_s X_s. \tag{22}$$

Here n_s denotes the natural gradient without a learning-rate factor; the corresponding descent direction is $-n_s$.

Proof. By the definition of F_s and $\psi_s(a) = J_s^\top(\mathbf{e}_a - p_s)$,

$$\begin{aligned} F_s &= \sum_a p_s(a)\psi_s(a)\psi_s(a)^\top \\ &= J_s^\top\left[\sum_a p_s(a)(\mathbf{e}_a - p_s)(\mathbf{e}_a - p_s)^\top\right]J_s. \end{aligned} \tag{23}$$

The matrix inside the brackets is

$$\begin{aligned} &\sum_a p_s(a)(\mathbf{e}_a - p_s)(\mathbf{e}_a - p_s)^\top \\ &= \sum_a p_s(a)\mathbf{e}_a\mathbf{e}_a^\top - \left(\sum_a p_s(a)\mathbf{e}_a\right)p_s^\top \\ &\quad - p_s\left(\sum_a p_s(a)\mathbf{e}_a\right)^\top + \left(\sum_a p_s(a)\right)p_s p_s^\top \\ &= \mathrm{Diag}(p_s) - p_s p_s^\top \\ &= G_s. \end{aligned} \tag{24}$$

Hence,

$$F_s = J_s^\top G_s J_s. \tag{25}$$

By definition, the parameter-space natural gradient is

$$n_s := F_s^\dagger g_s, \tag{26}$$

where the Moore–Penrose pseudoinverse allows for a possibly singular conditional Fisher matrix. Using $F_s = J_s^\top G_s J_s$ and $g_s = J_s^\top G_s X_s$, we obtain

$$n_s = (J_s^\top G_s J_s)^\dagger J_s^\top G_s X_s. \tag{27}$$

□

Lemma 3 (Fisher–Jacobian projection)**.** *Let*

$$B_s := G_s^{1/2} J_s, \qquad y_s := G_s^{1/2} X_s. \tag{28}$$

Then

$$\begin{aligned} &X_s^\top G_s J_s (J_s^\top G_s J_s)^\dagger J_s^\top G_s X_s \\ &\quad = \left\| P_{\mathrm{Range}(B_s)} y_s \right\|_2^2 \le \|y_s\|_2^2 = X_s^\top G_s X_s, \end{aligned} \tag{29}$$

where $P_{\mathrm{Range}(B_s)}$ *denotes the orthogonal projector onto* $\mathrm{Range}(B_s)$.

Equality holds if and only if

$$G_s^{1/2} X_s \in \mathrm{Range}(G_s^{1/2} J_s). \tag{30}$$

Proof. Since $J_s^\top G_s J_s = B_s^\top B_s$ and $J_s^\top G_s X_s = B_s^\top y_s$,

$$\begin{aligned} &X_s^\top G_s J_s (J_s^\top G_s J_s)^\dagger J_s^\top G_s X_s \\ &\quad = y_s^\top B_s (B_s^\top B_s)^\dagger B_s^\top y_s \\ &\quad = y_s^\top P_{\mathrm{Range}(B_s)} y_s \\ &\quad = \left\| P_{\mathrm{Range}(B_s)} y_s \right\|_2^2. \end{aligned} \tag{31}$$

The middle matrix is the orthogonal projector onto $\mathrm{Range}(B_s)$. Indeed, if $B_s = U_r \Sigma_r V_r^\top$ is a compact SVD, then

$$B_s (B_s^\top B_s)^\dagger B_s^\top = U_r U_r^\top = P_{\mathrm{Range}(B_s)}.$$

An orthogonal projection cannot increase the Euclidean norm, giving the inequality. Equality holds exactly when $y_s \in \mathrm{Range}(B_s)$. □

We now prove a more general statement that includes Proposition 4.2 as the locally realizable case.

Proposition A.1 (OPD–NPG gradient pairing)**.** *Fix a prefix* s*, and let*

$$g_s = \nabla_\theta L_s(\theta), \qquad n_s = F_s^\dagger g_s$$

denote the ordinary OPD parameter gradient and its parameter-space natural gradient, respectively. Then

$$\langle g_s, n_s \rangle = \left\| P_{\mathrm{Range}(G_s^{1/2} J_s)} G_s^{1/2} X_s \right\|_2^2 \le E_s. \tag{32}$$

Moreover, equality holds if and only if

$$G_s^{1/2} X_s \in \mathrm{Range}(G_s^{1/2} J_s). \tag{33}$$

In particular, under this local realizability condition,

$$\langle g_s, n_s \rangle = E_s. \tag{34}$$

Proof. By Lemmas 1 and 2,

$$\begin{aligned}\langle g_s, n_s\rangle &= g_s^\top F_s^\dagger g_s \\ &= X_s^\top G_s J_s (J_s^\top G_s J_s)^\dagger J_s^\top G_s X_s. \end{aligned} \tag{35}$$

Applying Lemma 3 gives

$$\langle g_s, n_s\rangle = \left\| P_{\mathrm{Range}(G_s^{1/2} J_s)} G_s^{1/2} X_s \right\|_2^2 \le \|G_s^{1/2} X_s\|_2^2. \tag{36}$$

The upper bound is

$$\begin{aligned}\|G_s^{1/2} X_s\|_2^2 &= X_s^\top G_s X_s \\ &= X_s^\top \left[\mathrm{Diag}(p_s) - p_s p_s^\top\right] X_s \\ &= \sum_a p_s(a) X_s(a)^2 - (p_s^\top X_s)^2 \\ &= \sum_a p_s(a) X_s(a)^2 \\ &= \mathrm{Var}_{a\sim p_s}[A_s(a)] = E_s, \end{aligned} \tag{37}$$

where $p_s^\top X_s = 0$ by construction.

Finally, by Lemma 3, the projection preserves the full norm if and only if

$$G_s^{1/2} X_s \in \mathrm{Range}(G_s^{1/2} J_s).$$

Under this condition, $\langle g_s, n_s\rangle = E_s$, which proves Proposition A.1. □

Equal Fisher budget interpretation. Proposition 4.2 also characterizes the first-order progress available under a fixed local change in the student policy. Consider

$$\max_{\Delta\in\mathbb{R}^d} -g_s^\top \Delta \qquad \text{subject to} \qquad \frac{1}{2}\Delta^\top F_s \Delta \le \epsilon.$$

Let $B_s = G_s^{1/2} J_s$ and $y_s = G_s^{1/2} X_s$. From the definitions above,

$$F_s = B_s^\top B_s, \qquad g_s = B_s^\top y_s.$$

Hence $g_s \in \mathrm{Range}(B_s^\top) = \mathrm{Range}(F_s)$. Directions in $\mathrm{Null}(F_s)$ neither consume Fisher budget nor change the first-order objective.

Since $g_s \in \mathrm{Range}(F_s)$,

$$-g_s^\top \Delta = -\left(F_s^{\dagger/2} g_s\right)^\top \left(F_s^{1/2}\Delta\right) \tag{38}$$

$$\le \sqrt{g_s^\top F_s^\dagger g_s}\sqrt{\Delta^\top F_s \Delta} \tag{39}$$

$$\le \sqrt{2\epsilon\, g_s^\top F_s^\dagger g_s}. \tag{40}$$

For $g_s \neq 0$, the bound is attained by

$$\Delta_s^\star = -\sqrt{\frac{2\epsilon}{g_s^\top F_s^\dagger g_s}} F_s^\dagger g_s,$$

which satisfies $\frac{1}{2}\Delta_s^{\star\top} F_s \Delta_s^\star = \epsilon$. Therefore,

$$\max_{\frac{1}{2}\Delta^\top F_s\Delta\le\epsilon} -g_s^\top \Delta = \sqrt{2\epsilon\, g_s^\top F_s^\dagger g_s}.$$

Under the local realizability condition of Proposition 4.2, $g_s^\top F_s^\dagger g_s = E_s$, giving

$$\max_{\frac{1}{2}\Delta^\top F_s\Delta\le\epsilon} -g_s^\top \Delta = \sqrt{2\epsilon E_s}.$$

When $E_s = 0$, the maximum first-order reduction is zero.

A.2 Proof of Proposition 4.1

Proposition (SNR-guided local update scaling)**.** *Fix a stored prefix c_t and consider the local reverse-KL objective as a function of the student logits,*

$$\mathcal{L}_t(z_t) = D_{\mathrm{KL}}(\mathrm{softmax}(z_t)\|q_t).$$

Let $u_t = \nabla_{z_t}\mathcal{L}_t(z_t)$, and let $\widehat{u}_t = u_t + \xi_t$ be an unbiased estimator of u_t, where $\mathbb{E}[\xi_t] = 0$ and $0 < \mathbb{E}\|\xi_t\|_2^2 = \sigma_t^2 < \infty$. Define the local gradient signal-to-noise ratio, as in Section 4.2, as

$$\Gamma_t = \frac{\|u_t\|_2^2}{\mathbb{E}\|\xi_t\|_2^2}. \tag{41}$$

Let $\beta > 0$ be a common smoothness upper bound over the stored prefixes under consideration, and suppose $\mathcal{L}_t$ is β-smooth. Let $\eta > 0$. For a deterministic weight w and the scalar-rescaled update

$$\Delta_t(w) = -\eta w\widehat{u}_t, \qquad w \geq 0,$$

the expected smoothness upper bound is minimized at

$$w_t^\star = \frac{1}{\beta\eta}\frac{\Gamma_t}{1+\Gamma_t}. \tag{42}$$

For fixed β and η, $w_t^\star$ is monotonically increasing and saturating in the gradient SNR.

Proof. Let $\mathcal{L}_t$ denote the local reverse-KL objective, with

$$u_t = \nabla_{z_t}\mathcal{L}_t(z_t).$$

By β-smoothness,

$$\mathcal{L}_t(z_t + \Delta_t(w)) \leq \mathcal{L}_t(z_t) + \langle u_t, \Delta_t(w)\rangle + \frac{\beta}{2}\|\Delta_t(w)\|_2^2. \tag{43}$$

Substituting $\Delta_t(w) = -\eta w\widehat{u}_t$ and taking expectations gives

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}_t(z_t + \Delta_t(w))\right] \leq \mathcal{L}_t(z_t) &- \eta w\,\mathbb{E}\left[\langle u_t, \widehat{u}_t\rangle\right] \\ &+ \frac{\beta\eta^2 w^2}{2}\mathbb{E}\|\widehat{u}_t\|_2^2.\end{aligned} \tag{44}$$

Since $\widehat{u}_t = u_t + \xi_t$ and $\mathbb{E}[\xi_t] = 0$,

$$\mathbb{E}\left[\langle u_t, \widehat{u}_t\rangle\right] = \|u_t\|_2^2,$$

and

$$\begin{aligned}\mathbb{E}\|\widehat{u}_t\|_2^2 &= \mathbb{E}\|u_t + \xi_t\|_2^2 \\ &= \|u_t\|_2^2 + 2\left\langle u_t, \mathbb{E}[\xi_t]\right\rangle + \mathbb{E}\|\xi_t\|_2^2 \\ &= \|u_t\|_2^2 + \mathbb{E}\|\xi_t\|_2^2.\end{aligned} \tag{45}$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}_t(z_t + \Delta_t(w))\right] \leq \mathcal{L}_t(z_t) &- \eta w\|u_t\|_2^2 \\ &+ \frac{\beta\eta^2 w^2}{2}\left(\|u_t\|_2^2 + \mathbb{E}\|\xi_t\|_2^2\right).\end{aligned} \tag{46}$$

The w-dependent part of the upper bound is

$$\phi_t(w) = -\eta w\|u_t\|_2^2 + \frac{\beta\eta^2 w^2}{2}\left(\|u_t\|_2^2 + \mathbb{E}\|\xi_t\|_2^2\right).$$

Differentiating gives

$$\phi_t'(w) = -\eta\|u_t\|_2^2 + \beta\eta^2 w\left(\|u_t\|_2^2 + \mathbb{E}\|\xi_t\|_2^2\right).$$

Setting $\phi_t'(w) = 0$ yields

$$\begin{aligned}w_t^\star &= \frac{1}{\beta\eta}\frac{\|u_t\|_2^2}{\|u_t\|_2^2 + \mathbb{E}\|\xi_t\|_2^2} \\ &= \frac{1}{\beta\eta}\frac{\Gamma_t}{1+\Gamma_t}.\end{aligned} \tag{47}$$

The quadratic is strictly convex, so this stationary point is the unique minimizer over $w \geq 0$. Finally, $\Gamma/(1+\Gamma)$ is monotonically increasing for $\Gamma \geq 0$ and converges to 1 as $\Gamma \to \infty$. Hence the oracle scale saturates at $1/(\beta\eta)$. □

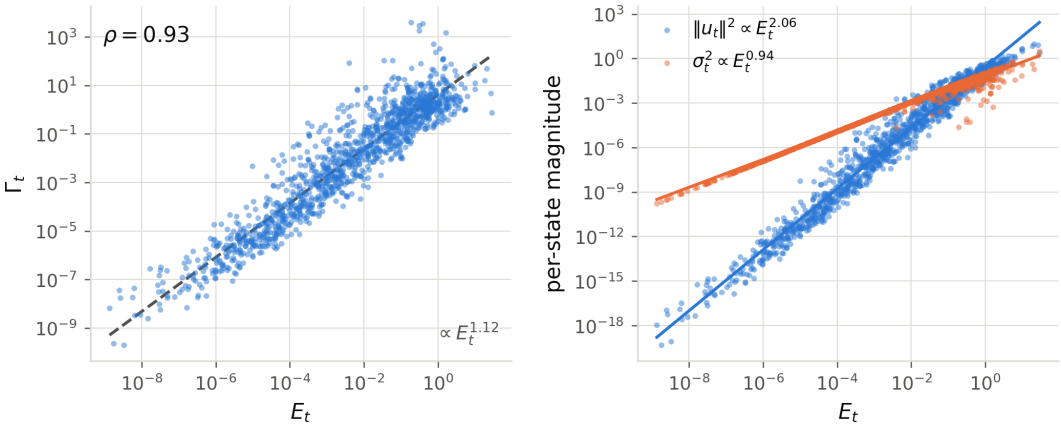

Figure 6: **Offline diagnostic of RKL variance and finite-sample gradient reliability.** We collect 1280 student-visited prefix states and store the full-vocabulary student and teacher conditional distributions, from which the RKL variance, squared logit-gradient signal, and sampling variance of the K=16 one-step resampling estimator are computed offline. **Left:** RKL variance is strongly rank-correlated with gradient SNR ($\rho = 0.93$), with the approximate scaling $\Gamma_t \propto E_t^{1.12}$. **Right:** the squared gradient signal scales approximately as $\|u_t\|_2^2 \propto E_t^{2.06}$, while the estimator variance scales as $\sigma_t^2 \propto E_t^{0.94}$.

B Additional Results and Analyses

B.1 RKL Variance and Gradient Reliability

Proposition 4.1 motivates increasing but bounded priority as a function of logit-gradient reliability. We next examine whether RKL variance provides a practical surrogate for this reliability.

For a stored prefix c_t, let p_t, q_t, and A_t be defined as in Section 4.2. The logit gradient in Proposition 4.1 has the closed form

$$u_t = \nabla_{z_t} D_{\mathrm{KL}}(p_t \| q_t) = \mathbb{E}_{a \sim p_t}\left[A_t(a)(\mathbf{e}_a - p_t)\right], \tag{48}$$

where $z_t = z_\theta(c_t)$ and $\mathbf{e}_a$ is the one-hot vector associated with action a.

Let $\widehat{u}_t$ denote the unbiased K-sample estimator corresponding to one-step resampling. Conditioned on the current model and prefix, its SNR, as defined in Section 4.2, is

$$\Gamma_t = \frac{\|u_t\|_2^2}{\sigma_t^2}, \qquad \sigma_t^2 := \operatorname{tr} \operatorname{Cov}(\widehat{u}_t). \tag{49}$$

To make this diagnostic tractable, we perform it offline on a small subset of stored prefixes rather than as part of training. Specifically, we collect 1280 student-visited prefix states, evaluate and store the full-vocabulary student and teacher conditional distributions, and then compute E_t, the logit-gradient signal, and the sampling variance offline.

As shown in Figure 6, exact RKL variance E_t exhibits a strong positive rank correlation with Γ_t across the evaluated prefixes. Higher-RKL-variance positions therefore tend to provide more reliable finite-sample gradient estimates, making the ordering induced by E_t broadly consistent with the reliability ordering favored by Proposition 4.1.

The signal–noise decomposition helps explain this association. Toward lower RKL variance, the squared gradient signal decreases more rapidly than the estimator variance. Although both quantities become smaller, their ratio also decreases. Thus, low-E_t positions exhibit not only weaker population-level optimization signal, as characterized by Proposition 4.2, but also lower relative reliability under finite-K estimation.

Although E_t is strongly correlated with gradient SNR, this relationship does not provide a calibrated numerical mapping between the two quantities. As illustrated by the empirical distribution of $\widehat{E}_t^K$

(Section B.2), its magnitude varies substantially across positions and training updates. Directly substituting E_t into the oracle weighting function would therefore require an additional choice of calibration.

During training, we estimate RKL variance with $\widehat{E}_t^K$ using the same candidate tokens and reverse-KL signals already available from one-step resampling, requiring no additional model evaluations. We use this estimate to recover a coarse priority ordering rather than treating its raw magnitude as a calibrated reliability score.

Implications for bounded priorities. The oracle rule in Equation (5) suggests two qualitative properties for practical prioritization: priority should increase with gradient reliability, while its amplification should remain bounded. Combined with the observed RKL-variance–SNR association, this motivates our bounded two-level rule in Equation (7). The rule preserves the coarse priority ordering suggested by E_t while avoiding direct dependence on its potentially unstable magnitude. We additionally retain nonzero weight on the low-variance group because a small finite-K estimate does not imply that the underlying local learning signal is exactly zero.

Remark (Logit-space reliability)**.** *The reliability diagnostic is evaluated in logit space at each fixed prefix. Let $J_t = \partial z_t / \partial \theta$ denote the logit Jacobian. The corresponding parameter-space gradient and estimation error satisfy*

$$g_t = J_t^\top u_t, \qquad \widehat{g}_t - g_t = J_t^\top (\widehat{u}_t - u_t).$$

Thus, with $\Sigma_t = \mathrm{Cov}(\widehat{u}_t)$, the parameter-space sampling variance is

$$\mathbb{E}\|\widehat{g}_t - g_t\|_2^2 = \mathrm{tr}\left(J_t J_t^\top \Sigma_t\right),$$

and therefore additionally depends on the local full-model Jacobian geometry.

We use logit space because E_t, u_t, and the sampling variance of the K-sample estimator are determined entirely by the student and teacher next-token distributions at the fixed prefix and can therefore be computed exactly from the stored full-vocabulary distributions. Computing the corresponding parameter-space moments additionally requires full-model Jacobian information at each prefix and is substantially more expensive. The smoothness argument of Proposition 4.1 also applies in parameter space, yielding the same increasing and saturating form with

$$\Gamma_t^\theta = \frac{\|g_t\|_2^2}{\mathrm{tr}(J_t J_t^\top \Sigma_t)}$$

and the corresponding parameter-space smoothness constant.

B.2 Distribution of Estimated RKL Variance

We examine the empirical distribution of the one-step resampling RKL-variance estimator $\widehat{E}_t^K$. As shown in Figure 7, the distribution contains substantial mass at zero and near zero. This behavior is partly a finite-sample effect: $\widehat{E}_t^K = 0$ whenever the K sampled candidates yield identical reverse-KL signals, which can occur frequently for peaked next-token distributions and finite K. Therefore, a zero estimated RKL variance does not necessarily imply that the exact E_t is zero.

Among nonzero estimates, $\widehat{E}_t^K$ spans several orders of magnitude and exhibits a pronounced right tail. Directly using its raw magnitude as a continuous optimization weight can therefore make the update overly sensitive to a small number of extreme estimates. We instead use a bounded two-level weighting rule, which preserves the coarse distinction between positions with weaker and stronger remaining optimization signal while limiting sensitivity to extreme variance estimates.

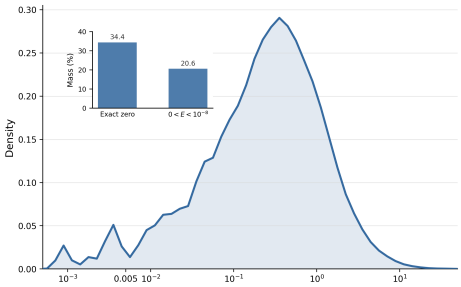

Figure 7: Distribution of the one-step resampling RKL-variance estimator $\widehat{E}_t^K$. A substantial fraction of positions have zero or near-zero estimated variance, while the nonzero values span several orders of magnitude and exhibit a long right tail. The inset reports the probability mass at exactly zero and in $(0, 10^{-8})$.

Table 4: Evaluation on additional model families. AMC23, MATH-500, and MMLU-Redux are evaluated with Avg@16, Avg@4, and Avg@1, respectively. REVO is evaluated at Step 50, while all other methods are evaluated at Step 200. The highest score in each column is shown in bold.

	Llama 3.2 3B			Gemma 3 4B		
Method	**AMC23**	**MATH-500**	**MMLU-Redux**	**AMC23**	**MATH-500**	**MMLU-Redux**
Base	0.3	0.3	6.3	0.3	0.5	11.5
Sampled-Token OPD	1.9	3.4	12.5	13.4	34.7	53.4
EOPD	1.1	2.4	6.7	12.8	33.9	27.1
FiRe	1.7	2.6	9.2	12.5	33.1	48.6
REVO	**6.4**	**8.9**	**28.3**	**16.9**	**37.7**	**54.1**

B.3 Additional Model Families

We further evaluate REVO on Llama 3.2 and Gemma 3 to examine whether its rollout efficiency gains extend beyond Qwen3. For Llama, we use `Llama-3.2-3B` as the base student and `Llama-3.2-3B-Instruct` as the teacher. For Gemma, we use `gemma-3-4b-pt` as the base student and `gemma-3-4b-it` as the teacher. Evaluations on AIME24 and AIME25 yield accuracies close to zero for these model pairs and provide limited separation between methods. We therefore focus on AMC23 and MATH-500 for mathematical reasoning and MMLU-Redux for cross-domain evaluation.

As shown in Table 4, REVO achieves the highest accuracy on all three reported benchmarks for both model families, outperforming sampled-token OPD, EOPD, and FiRe. Notably, REVO uses only 50 rollout iterations, while all other methods are trained for 200 iterations. These results show that the benefits of REVO extend beyond Qwen3 and remain effective across different model families.

Chat template and answer format. For each model family, we use the chat template associated with the corresponding teacher checkpoint for both training and evaluation. Across all families, we use the same boxed-answer format, where each prompt instructs the model to place the final answer in `\boxed{}`. The same template and answer format are used consistently for training and evaluation.

B.4 Complete Training Procedure

We present the complete training procedure of REVO in Algorithm 1. The algorithm combines stabilized prefix correction, one-step resampling from the current policy, and reweighting guided by RKL variance with weights normalized to have mean one. Following Equation (4), the correction weights and reverse-KL signals are treated as constants in the stop-gradient surrogate. In Algorithm 1, we add a batch index i to all per-position quantities, writing $c_{i,t}$ for the prefix of response i at position t and $A_{i,t}^{(k)} := A_{i,t}(a_{i,t}^{(k)})$.

B.5 One-Step Resampling in OPD

We further isolate the effect of one-step resampling in OPD without rollout reuse. Each training iteration uses freshly generated student rollouts, so there is no policy staleness from repeated off-policy updates.

Under the usual rollout setting with $T = 1$ and top-$p = 1$, rollout tokens are sampled directly from the student distribution defining the reverse-KL objective. In this case, one-step resampling mainly improves the finite-sample conditional gradient estimate by averaging multiple current-policy samples.

We also generate training rollouts using the evaluation sampling setting, $T = 0.7$ and top-$p = 0.95$. This changes the rollout-token distribution while leaving the reverse-KL objective unchanged. As shown in Figure 8, sampled-token OPD degrades substantially under this altered sampling distribution, while one-step resampling largely preserves the performance obtained with $T = 1$ and top-$p = 1$.

Algorithm 1 Training procedure for REVO

Require: Student policy P_θ, teacher P_T, prompt dataset $\mathcal{D}$
Require: Inner updates N, resampling size K, prefix cap C, RKL-variance threshold δ, weighting coefficient τ

for each rollout iteration **do**
  Freeze the behavior policy $P_o \leftarrow P_\theta$
  Sample prompts from $\mathcal{D}$ and generate rollout batch $\mathcal{B}$ using P_o
  Store behavior log-probabilities for all generated tokens
  for $n = 1, \ldots, N$ **do**
    Evaluate the current student and teacher on all stored prefixes in $\mathcal{B}$
    for each valid position $(i, t) \in \mathcal{M}_\mathcal{B}$ **do**
      Compute the prefix correction $G_{i,<t}$ using Equation (3), with $G_{i,<1} = 1$
      Sample

$$a_{i,t}^{(1)}, \ldots, a_{i,t}^{(K)} \overset{\text{i.i.d.}}{\sim} P_\theta(\cdot \mid c_{i,t})$$

      Compute the token-level reverse-KL signals $A_{i,t}^{(k)} = \log P_\theta(a_{i,t}^{(k)} \mid c_{i,t}) - \log P_T(a_{i,t}^{(k)} \mid c_{i,t})$ for $k = 1, \ldots, K$
      Estimate the local RKL variance $\widehat{E}_{i,t}^{K}$ using Equation (6)
      Compute the raw RKL-variance weight $\widehat{w}_{i,t}$ using Equation (7)
    end for
    Normalize weights over valid positions

$$w_{i,t} = \frac{\widehat{w}_{i,t}}{\frac{1}{|\mathcal{M}_\mathcal{B}|} \sum_{(j,u) \in \mathcal{M}_\mathcal{B}} \widehat{w}_{j,u}}$$

    Construct the reweighted resampling objective

$$\widehat{\mathcal{L}} = \frac{1}{|\mathcal{M}_\mathcal{B}|} \sum_{(i,t) \in \mathcal{M}_\mathcal{B}} \frac{1}{K} \sum_{k=1}^{K} \mathrm{sg}\Big[G_{i,<t}\, w_{i,t}\, A_{i,t}^{(k)}\Big] \log P_\theta\left(a_{i,t}^{(k)} \mid c_{i,t}\right)$$

    Update

$$\theta \leftarrow \theta - \eta \nabla_\theta \widehat{\mathcal{L}}$$

  end for
end for

These results show two effects of one-step resampling in OPD. It improves finite-sample gradient estimation when rollout tokens are sampled directly from the student distribution, and it additionally corrects current-token mismatch when the rollout sampling distribution is altered. Under rollout reuse, the same resampling step addresses current-token mismatch caused by the student moving away from the behavior policy, while prefix correction separately accounts for the shift in stored prefixes.

C Experimental Details and Ablations

C.1 Implementation Details

We provide the implementation details of REVO in this section. All experiments are conducted on $4\times$ NVIDIA A100-SXM4-80GB GPUs. We use the DAPO-Math-17K dataset and convert the prompts to the boxed template. The same prompt template is used for evaluation to ensure consistency between training and evaluation. Our codebase builds on the implementation of Li et al. (2026a), and we apply the EOS-alignment method of Yang et al. (2026c) to address termination-token mismatch for all methods in this paper, including the baselines. We summarize the main implementation choices and hyperparameters in Table 5. Unless otherwise specified, all experiments use the same configuration. We set the prefix cap to $C = 4$ only as a numerical safeguard against rare extreme likelihood ratios.

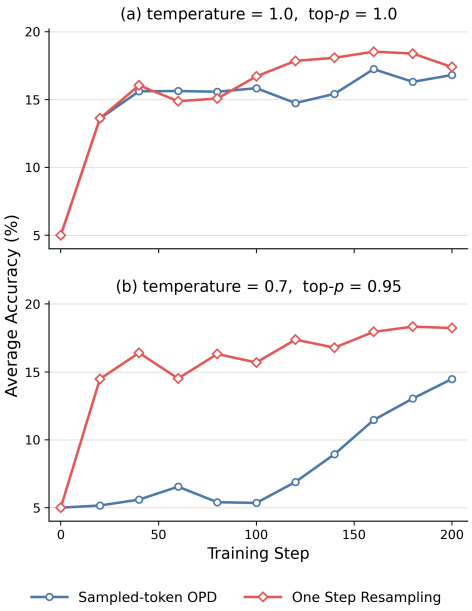

Figure 8: One-step resampling in OPD without rollout reuse. **(a)** Rollouts use $T = 1$, top-$p = 1$. **(b)** Rollouts use the evaluation setting $T = 0.7$, top-$p = 0.95$. Resampling gives a modest gain in (a) and substantially mitigates the degradation in (b). Results are Avg@16 over AIME24, AIME25, and AMC23.

Ablation details. We provide additional implementation details for the alternative correction and weighting rules evaluated in Table 3.

PPO-style clipped importance sampling. As an alternative current-token correction, we use the behavior policy that generated each rollout as P_{o} and form the token importance ratio

$$r_t(\theta) = \frac{P_\theta(a_t \mid c_t)}{P_{\mathrm{o}}(a_t \mid c_t)}.$$

We apply a dual-clipped PPO surrogate to the same token-level reverse-KL signal used by OPD, with clipping interval $[0.8, 1.2]$ and dual-clip constant $c_{\mathrm{dual}} = 3$ applied to positive reverse-KL signals, which correspond to negative advantages under the PPO convention. This baseline corrects only the current-token ratio and does not use prefix correction.

Alternative RKL-variance weighting. For the $\sqrt{\widehat{E}^K}$ variant, we directly use $\sqrt{\widehat{E}_t^K}$ as the token weight and normalize the weights to have mean one over valid positions, without additional clipping. For the SNR-inspired saturating rule $\widehat{E}_t^K/(\widehat{E}_t^K + c)$, we use $c = 0.25$. This value approximately matches the median of the mean RKL variance across training iterations in the default run. In comparison, $c = 1$ is poorly matched to the observed scale because most values of $\widehat{E}_t^K$ are substantially smaller than one.

C.2 Prompt Template

We use the following user prompt across all model families for both training and evaluation. For each family, it is formatted using the chat template associated with the corresponding teacher.

```
{problem} Please reason step by step, and put your final answer
within \boxed{}.
```

Table 5: Implementation details and hyperparameters used in the main experiments.

Category	Hyperparameter	Setting
Method	Candidate samples K	16
	RKL-variance threshold δ	0.005
	Reweighting coefficient τ	0.75
	Weight normalization	Mean 1 over valid tokens
	Prefix-ratio cap C	4
Training	Training data	DAPO-Math-17K
	Data order	Sequential (`shuffle=False`)
	Outer training steps	50
	Prompts per step	8
	Responses per prompt	4
	Inner updates per rollout	10
	Maximum model/token length	8,192
	Student rollout sampling	temperature $= 1.0$, top-$p = 1.0$
	Teacher temperature	1.0
	Optimizer	AdamW
	Learning rate	1×10^{-6}
	Weight decay	0.01
	Gradient clipping	1.0
	Precision	FP32
	Thinking mode	Disabled
Evaluation	Temperature	0.7
	Top-p	0.95
	Maximum new tokens	8,192
	Prompt template	Boxed answer
Compute	GPUs	$4\times$ NVIDIA A100-SXM4-80GB
	FSDP world size	4
	Tensor parallel size	1

C.3 Hyperparameter Validation

We further evaluate the sensitivity of REVO to the hyperparameters used for RKL-variance reweighting. Experiments are conducted with a Qwen3-1.7B Base student distilled from a Qwen3-4B teacher. We vary the RKL-variance threshold δ and the reweighting coefficient τ, and report Avg@16 over AIME24, AIME25, and AMC23.

As shown in Table 7, REVO remains effective across a range of hyperparameter choices. Increasing τ from 0.6 to moderate values of 0.7 and 0.75 improves performance, while increasing it further to 0.8 leads to some degradation. This suggests that overly aggressive prioritization can concentrate optimization too strongly on the high-variance positions. Performance is also relatively robust to the RKL-variance threshold δ. The default value $\delta = 0.005$ achieves the best result among the tested settings, while substantially different thresholds remain competitive.

We further ablate the number of candidate tokens K used for one-step resampling. To isolate its effect, we disable the other components of REVO and vary only K. Results are shown in Table 6.

Performance improves slightly as K increases, from 17.6 at $K = 1$ to 18.1 at $K = 64$, while remaining stable across the intermediate choices. This indicates that one-step resampling is not highly sensitive to K, with larger sample sizes providing diminishing gains.

Table 6: Effect of the resampling size K for a Qwen3-1.7B Base student distilled from a Qwen3-4B teacher, evaluated with Avg@16 over AIME24, AIME25, and AMC23, with other REVO components disabled.

K	1	4	16	64
Avg@16	17.6	17.8	17.8	18.1

C.4 Alternative Prioritization Signals

We compare RKL variance with two alternative signals computed from the same K candidate tokens

Table 7: Hyperparameter sensitivity of RKL-variance reweighting for a Qwen3-1.7B Base student distilled from a Qwen3-4B teacher. Results are Avg@16 over AIME24, AIME25, and AMC23. † denotes the default configuration.

Varying τ ($\delta = 0.005$)		**Varying** δ ($\tau = 0.75$)	
τ	**Avg@16**	δ	**Avg@16**
0.60	18.8	0.001	19.3
0.70	**19.7**	$0.005^{\dagger}$	**19.6**
$0.75^{\dagger}$	19.6	0.050	18.7
0.80	19.1	0.200	18.9

Table 8: Comparison of token-level prioritization signals for a Qwen3-1.7B Base student distilled from a Qwen3-4B teacher. The high-weight fraction is the fraction of valid token positions assigned weight τ before normalization during training. Results are Avg@16 over AIME24, AIME25, and AMC23.

Prioritization signal	Threshold	High-weight fraction	Avg@16
Uniform weighting	–	–	18.2
Sampled KL	0.02	34.0%	17.3
Sampled entropy	0.27	32.6%	16.5
RKL variance (REVO)	0.005	32.8%	**19.6**

used for one-step resampling: sampled KL, $\bar{A}_t^K = \frac{1}{K}\sum_k A_t^{(k)}$, and sampled student entropy, $\widehat{H}_t = -\frac{1}{K}\sum_k \log p_t(a_t^{(k)})$.

Because the three signals have different numerical scales, a shared threshold would not give a fair comparison. We therefore use signal-specific thresholds chosen so that each signal assigns the higher weight to approximately the same fraction of positions as the default $\delta = 0.005$ of REVO, which is roughly one third. For each alternative signal, we first use a pilot run to estimate a threshold that matches this fraction, and then fix this threshold throughout the full training run. This gives thresholds of 0.02 for sampled KL and 0.27 for sampled student entropy. During full training, the resulting fractions of positions receiving the higher weight are close to one third for all three signals (Table 8). All other components, including the weighting coefficient and weight normalization, are unchanged. We use a Qwen3-1.7B Base student distilled from a Qwen3-4B teacher.

As shown in Table 8, RKL variance achieves the highest accuracy, while both alternative signals perform worse than uniform weighting despite nearly matched high-weight fractions. Sampled KL measures the average log probability ratio between the student and teacher, while student entropy depends only on the student distribution. RKL variance instead measures variation in this log ratio across candidate tokens, capturing the centered signal that contributes to the local reverse KL gradient.

C.5 Wall-Clock Time

We further report the wall-clock efficiency of REVO using Qwen3-1.7B Base as the student and Qwen3-4B as the teacher. On $4\times$ NVIDIA A100-SXM4-80GB GPUs, training REVO for 50 steps takes approximately 5.5 hours, whereas training standard OPD for 200 steps takes approximately 10 hours. As shown in Table 1, REVO substantially outperforms OPD under the same 50-step rollout budget and also surpasses OPD trained for 200 steps, demonstrating improved performance together with lower wall-clock cost.

We further profile the per-step runtime of REVO in Table 9. Even with 10 learner updates per rollout batch, student rollout generation remains the largest single component at 32.4% of the step. The additional updates add scoring and optimization cost, but each of them reuses the rollout rather than paying for autoregressive generation again. For comparison, a standard OPD step consists only of the rollout, one scoring pass, and one actor update, so generation accounts for roughly 70% of its wall-clock time. The breakdown makes the trade-off of REVO explicit: learner-side computation is

Table 9: Per-step wall-clock breakdown of REVO with 10 learner updates per rollout batch (U10), measured using a Qwen3-1.7B Base student and Qwen3-4B teacher on $4\times$ NVIDIA A100-SXM4-80GB GPUs.

Component	**Time (s)**	**Share**
Student rollout	128.7	32.4%
Initial student/teacher scoring	15.5	3.9%
9 additional student log-prob evaluations	52.1	13.1%
9 additional teacher scoring passes	33.8	8.5%
9 additional resampling signal constructions ($K = 16$)	14.2	3.6%
10 actor forward/backward/update passes	100.0	25.2%
Data movement, synchronization, and scheduling	52.5	13.2%
Total	**396.7**	**100%**

Table 10: Effect of rollout batch size for OPD with a Qwen3-1.7B Base student distilled from a Qwen3-4B teacher. Relative rollout budget is normalized to the default REVO setting. Results are Avg@16 over AIME24, AIME25, and AMC23.

Method	Rollout iterations	Prompts per iteration	Relative rollout budget	Avg@16
REVO	50	8	$1\times$	**19.6**
OPD	200	8	$4\times$	16.8
OPD	200	16	$8\times$	17.5
OPD	200	32	$16\times$	18.4

spent to extract more optimization progress from each generated rollout, and the share of time spent generating drops from about 70% to about 32%.

C.6 Effect of Rollout Batch Size

We further examine whether increasing the rollout batch size improves OPD. We increase the number of prompts per iteration from 8 to 16 and 32 and train each configuration for 200 rollout iterations. Relative rollout budget is normalized to the default REVO setting, which uses 8 prompts per iteration for 50 rollout iterations.

As shown in Table 10, increasing the rollout batch size steadily improves OPD. However, even with a $16\times$ relative rollout budget, OPD remains below REVO trained with the default rollout budget.