# RANDOM REFRESH IS ALL YOU FORGOT: RETHINKING STALENESS IN OFFLINE GRPO

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reinforcement learning for vision-language models is increasingly bottlenecked by rollout generation, motivating the reuse of stale, off-policy data. Existing methods address this problem through increasingly sophisticated selection, replay, reweighting, or clipping mechanisms, but rarely ask whether they outperform a much simpler alternative: refreshing a small random fraction of the rollout pool. We systematically compare eight methods that can be used to mitigate data staleness against a $\sim 15\%$ random-refresh control across eight multimodal benchmarks. On the primary model family, none of 64 method–benchmark cells exceeds random refresh by more than 1.2 accuracy points, with a median margin of $-4.2$ points; 31 of 34 cross-family spot checks show the same pattern. We then ask where staleness actually concentrates. Across 26 family–benchmark cells and four model families, visually dependent tokens exhibit substantially higher staleness rates than language-prior tokens, with a mean high-$|\mathrm{VD}|$/low-$|\mathrm{VD}|$ ratio of 9.1. Motivated by these findings, we construct a selective-refresh feasibility test that combines a small explorer with partial refresh by a larger trainee. The resulting system combines a small explorer with a larger trainee and refreshes groups with high visual dependence, supporting the first two findings while reducing the cost of full-size rollout sampling. Together, these results shift the focus from increasingly elaborate corrections of stale rollouts toward a simpler principle: maintaining freshness matters, and random refresh should be treated as a standard control whenever off-policy GRPO methods are evaluated.

## 1 INTRODUCTION

Post-training VLMs with GRPO-style reinforcement learning (Shao et al., 2024) is dominated by the cost of generating rollouts: sampling is memory-bound and slow, and on-policy GRPO (Yu et al., 2026a) re-samples after every update. Reusing a pool over many epochs amortizes this cost, but the policy keeps moving while the pool does not, so samples an earlier policy generated increasingly disagree with the policy training on them, a drift we call "staleness". Re-sampling the whole pool restores agreement but gives back the saving reuse was meant to buy. Efficient post-training therefore needs an answer to one question: *how to mitigate that staleness at the lowest possible cost.*

Existing work instead adjusts how already-sampled rollouts participate in the update. One line selects which tokens or samples get gradient, replay, or refresh by an informativeness signal (Wang et al., 2026; Ye et al., 2026; Yoo et al., 2026). A second bounds how much gradient a stale sample already in the batch contributes (Zheng et al., 2026; Xi et al., 2026; Gao et al., 2025; Tian et al., 2026). Both lines decide how stale rollouts are used; neither decides whether they should be replaced.

Using rollouts effectively and deciding whether to replace them are distinct questions: staleness reflects a distributional gap from the current policy, whereas the value of refresh depends on whether replacing samples improves training. For VLMs, the two also differ in granularity: diagnosis can be per token (Wang et al., 2026), while refresh operates per group (Yu et al., 2026a), so a drifted token does not mean its group is worth regenerating. This raises a direct question: how much of the benefit sought through sophisticated corrections comes instead from refreshing a small, uniformly random fraction of the pool? Such a control is informative because it supplies fresh data without any ranking, priority, or correction signal: a method that cannot beat it is not earning its additional machinery. To our knowledge, no published evaluation of these methods reports this comparison.



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.

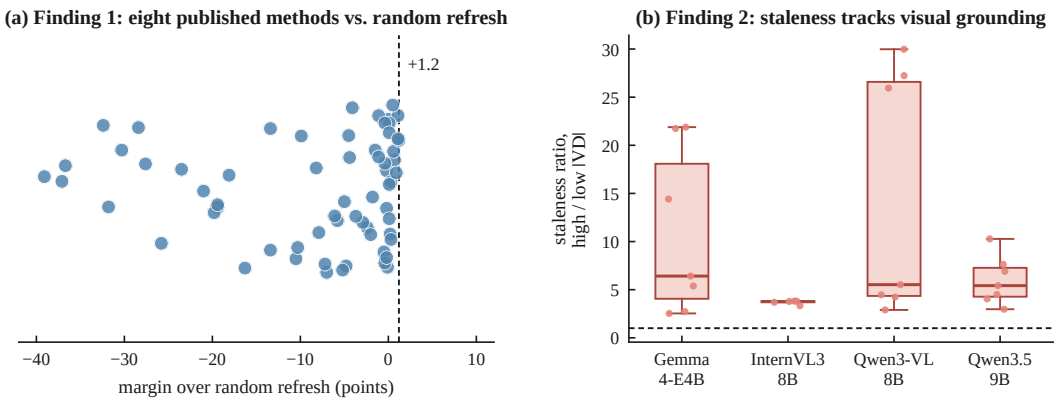


Figure 1: Two findings at a glance. **(a)** All 64 method × benchmark cells in the primary-family grid, the 8-benchmark main-text scope (Section 4), against a random $\sim$15% pool-refresh control; none exceed the $+1.2$-point bar used throughout. **(b)** Live re-evaluation: visually dependent tokens go stale more often than language-prior ones, per-cell ratio by family (Section 5).

As shown in Figure 1, running this comparison at scale reveals two empirical findings. We then utilize the second to motivate a selective-refresh feasibility test, examining whether the identified staleness pattern can be translated into an actionable refresh strategy.

**Finding 1 (an overlooked baseline).** We faithfully adapt eight recent methods from both lines to a single training and evaluation protocol, run on eight widely used public multimodal benchmarks against the random-refresh control (Section 4): the median margin is $-4.2$ points; on AI2D, every method loses between 8 and 30 points, with MMStar and ScienceQA losses also large but less uniform, and no method beats the control by more than 1.2 points on any of the eight benchmarks. Spot checks on three additional model families, 34 further judged cells, agree in 31 of 34, with three exceptions. These results suggest that random refresh should be included as a standard control when evaluating methods that can be used to mitigate data staleness.

**Finding 2 (a mechanism).** We re-evaluate token probabilities under the current trainee at every epoch and measure visual dependence by the change induced by masking the image. Across all 26 confirmation cells, covering four model families, high-$|\mathrm{VD}|$ tokens exhibit higher staleness rates than low-$|\mathrm{VD}|$ tokens, with a mean ratio of 9.1 (Section 5). This explains why existing selection signals miss staleness: entropy (Wang et al., 2026), advantage magnitude (Yoo et al., 2026), and clipping statistics (Xi et al., 2026; Gao et al., 2025) do not capture visual dependence, whereas random refresh introduces fresh samples without relying on any proxy signal.

**Method (a feasibility test).** Motivated by Finding 2, we test whether visual dependence can guide selective refresh at the rollout-group level. A small explorer samples the rollout pool once, and a larger trainee regenerates only the $\sim$15% of groups receiving the highest group-level scores each epoch. Across the complete 55-unit evaluation, the method obtains a positive margin over both random refresh and on-policy GRPO in 35 units; on ten representative settings, it exceeds on-policy GRPO by 0.8 to 8.4 points (Section 6). Its rollout cost per round is about 90% of a full-size pool, compared with 115% for random refresh, and measured end-to-end wall-clock is 20% to 96% of on-policy GRPO. We treat this as a feasibility study of selective refresh rather than as a replacement for standard on-policy GRPO training of the full-size trainee.

Our contributions are summarized as follows:

- We systematically compare eight methods that can be used to mitigate data staleness against a random-refresh control under a common protocol, with a full eight-benchmark evaluation on the primary model family and additional spot checks on three further families.
- We provide a token-level characterization of where staleness concentrates, showing through live training-time measurements that visually dependent tokens consistently exhibit higher staleness rates than language-prior tokens in all 26 cells across all four model families.
- We study a selective-refresh strategy that translates this diagnostic into a group-level rule, and characterize its accuracy–cost trade-off against random refresh and on-policy GRPO.

# 2 Related Work

## 2.1 Policy Lag and Rollout Reuse in GRPO

Autoregressive rollout generation is a major bottleneck in GRPO-style post-training (Shao et al., 2024): generation is sequential, memory-intensive, and must be repeated frequently under strictly on-policy training. Related forms of policy lag have long been studied in reinforcement learning. IMPALA (Espeholt et al., 2018) corrects actor–learner lag through importance weighting, while Noukhovitch et al. (2025) extend related off-policy corrections to asynchronous RLHF. These settings primarily address relatively short-lived policy lag during concurrent sampling and optimization, whereas repeated rollout reuse can preserve the same data across many policy updates. Complementary system-level approaches instead modify data collection itself: CoPRIS (Qu et al., 2025) combines concurrency-controlled partial rollouts with importance-sampling correction, while PODS (Xu et al., 2025) reduces sampling cost by down-sampling rollouts according to response variance. These studies motivate efficient departures from strict on-policy sampling, but leave a basic question open: when reused rollouts become stale, how much can simple refresh recover without sophisticated correction? Our work addresses this question using partial random refresh as an explicit control, separating the value of *correcting stale data* from that of *refreshing it*.

## 2.2 Selection and Prioritization Methods

A related line of work improves rollout reuse by prioritizing informative tokens or samples rather than treating all stored data equally. VPPO (Huang et al., 2026) and PGPO (Ye et al., 2026) use token-level visual dependence to reweight or gate advantages, while PEPO (Li et al., 2026) combines visual similarity and entropy for multimodal chain-of-thought. Entropy-based methods such as the 80/20 rule (Wang et al., 2026) and ERPO (Yu et al., 2026b) emphasize high-entropy tokens, whereas rollout-level advantage-prioritized replay (Yoo et al., 2026), following prioritized experience replay (Schaul et al., 2015), favors more informative rollouts. Related analyses further connect token entropy to RL dynamics in reasoning models (Cui et al., 2025). Despite different prioritization signals, these methods share a common focus: deciding which existing samples should contribute more to optimization. Our question is complementary: whether stale samples should remain in the pool at all. None of these signals directly measures policy drift, or whether replacing a stored rollout with fresh data would actually help more than simply keeping the original in place.

## 2.3 Distribution-Correction Methods

A second line of work mitigates off-policy drift by controlling how stale samples affect optimization. M2PO (Zheng et al., 2026) constrains importance-ratio drift with a second-moment trust region, BAPO (Xi et al., 2026) adapts clipping bounds asymmetrically, SAPO (Gao et al., 2025) replaces hard clipping with a soft gate, and Mu-GRPO (Tian et al., 2026) combines wider clipping with response-level vetoes and replay limits. Related methods such as DAPO (Yu et al., 2026a) instead oversample to remove zero-advantage groups. These approaches focus on *correcting* stale data once it enters optimization. Our work takes a different perspective: rather than correcting stale rollouts, we ask whether a small amount of fresh data can make such correction largely unnecessary.

# 3 Experimental Setup

**Research Questions.** Our experiments are organized around three questions. First, we test whether existing methods that can be used to mitigate data staleness outperform a simple random-refresh control under a common training and evaluation protocol (Section 4). Second, we diagnose where staleness concentrates by re-evaluating token probabilities under the evolving policy (Section 5). Third, we test whether the resulting diagnostic can support selective refresh in a small-explorer/large-trainee setting (Section 6). The first two studies use self-sampled pools to isolate within-training staleness, whereas the third adds an explorer–trainee capacity gap to reduce full-size rollout generation cost, as described in Section 6 and Table 3.

**Models and Evaluation Scope.** Findings 1–2 use self-sampled rollouts, where each model both generates and trains on its own data, isolating staleness from cross-model mismatch. InternVL3-

Table 1: Accuracy margin (percentage points) of eight methods that can be used to mitigate data staleness relative to the ∼15% random-refresh control on InternVL3-8B. Positive values favor the method; negative values favor random refresh.

| Method | MMStar | POPE | MME | MathV. | SciQA | DocVQA | LogicV. | AI2D |
|---|---|---|---|---|---|---|---|---|
| *Selection-based* | | | | | | | | |
| VPPO | −36.7% | −2.4% | −4.8% | −5.2% | −9.9% | 0.0% | −0.2% | −30.3% |
| PGPO | −37.1% | +1.1% | −4.5% | −7.0% | −13.4% | −0.1% | −1.5% | −25.8% |
| Ent-80/20 | −28.4% | +0.3% | −2.9% | −5.0% | −16.3% | −0.5% | +0.7% | −27.6% |
| Rollout-AP | −10.5% | +0.1% | +1.2% | −7.9% | −32.4% | −0.4% | −3.7% | −10.3% |
| *Distribution correction* | | | | | | | | |
| M2PO | −23.5% | −0.2% | +0.5% | −4.1% | −4.4% | −0.4% | −1.1% | −19.4% |
| Mu-GRPO | −13.4% | +0.6% | +0.1% | −5.8% | −21.0% | +0.1% | −1.1% | −19.8% |
| BAPO | −39.1% | +0.1% | +0.9% | −6.1% | −31.8% | −0.4% | −2.0% | −8.2% |
| SAPO | −1.8% | +1.1% | +0.2% | −7.2% | −19.4% | +0.3% | −0.2% | −18.1% |
| Median | −26.0% | +0.2% | +0.2% | −6.0% | −17.9% | −0.3% | −1.1% | −19.6% |

8B (Zhu et al., 2025) serves as the primary family for the exhaustive evaluation; cross-family analyses additionally cover Gemma (Gemma Team, 2026), Qwen3-VL (Qwen Team, 2025), and Qwen3.5 (Qwen Team, 2026) variants. For selective refresh, we evaluate five small-explorer/large-trainee configurations (1B–4B → 4B–9B) following Ren et al. (2026), summarized in Table 3.

**Benchmarks.** Finding 1 evaluates eight public multimodal benchmarks: MMStar (Chen et al., 2024), POPE (Li et al., 2023), MME (Fu et al., 2025), MathVista (Lu et al., 2024), ScienceQA (Lu et al., 2022), DocVQA (Mathew et al., 2021), LogicVista (Xiao et al., 2024), and AI2D (Kembhavi et al., 2016). Together, they cover perception, hallucination, document and diagram understanding, and mathematical and logical reasoning. Finding 2 evaluates staleness across seven benchmarks, while the selective-refresh study further includes BLINK (Fu et al., 2024), HallusionBench (Guan et al., 2024), and MMMU (Yue et al., 2024). Full benchmark coverage and evaluation details for the three studies are provided in Appendices A, B, and C, respectively.

**Training and Comparison Protocol.** Within each evaluation frame, all arms share the same initial rollout pool and training harness. For Finding 1, each method is compared against three references: *random refresh*, which uniformly re-samples ∼15% of rollout groups per epoch; *on-policy GRPO*, which re-samples the full pool each epoch; and a *frozen-pool* baseline with no refresh. Finding 2 measures staleness under the frozen-pool regime without intervention. For the selective-refresh study, we keep the ∼15% refresh budget fixed and compare targeted refresh against random refresh and on-policy GRPO. Unless otherwise specified, training uses 200 rollout groups for eight epochs, with group size $K$ determined by the model or explorer–trainee configuration (Table 3).

**Validity and Judging.** We judge comparisons only within matched evaluation frames, defined by the same evaluation slice and initial accuracy, and under identical seed budgets. A run is valid only if it completes evaluation without OOM, has nonzero optimization steps, keeps the exception rate below 10%, and passes the family-specific adapter-integrity check. A cell enters the record only when all compared arms satisfy these criteria on the same frame. We report the strongest seed for each arm under the same rule for methods and controls; the few frame-mismatched cases are explicitly flagged in the tables and discussed in the text. The full set of validity criteria, seed-level results, and the complete audit trail of judgment changes are all provided in Appendix C.

## 4 Finding 1: Published Methods Rarely Beat Random Refresh

We first ask whether existing methods that can be used to mitigate data staleness outperform the simplest freshness intervention: uniformly refreshing a small fraction of the rollout pool. Under the common protocol in Section 3, we compare eight representative methods against the ∼15%

random-refresh control on eight widely used multimodal benchmarks using InternVL3-8B as the primary model family for the exhaustive comparison.

**Compared Methods.** We evaluate two broad classes of approaches. Selection-based methods include VPPO (Huang et al., 2026), PGPO (Ye et al., 2026), Ent-80/20 (Wang et al., 2026), and Rollout-AP (Yoo et al., 2026), which prioritize or reweight tokens or rollouts using visual, entropy, or advantage signals. Distribution-correction methods include M2PO (Zheng et al., 2026), BAPO (Xi et al., 2026), SAPO (Gao et al., 2025), and Mu-GRPO (Tian et al., 2026), which instead control how strongly stale samples affect optimization through trust regions, adaptive clipping, soft gating, or explicit constraints on how often each stored sample may be replayed during training.

**Main Result.** Table 1 reports, for each method and benchmark on the primary family (InternVL3-8B), the margin of each method's strongest seed over the random-refresh control. Across 64 valid cells, the median margin is $-4.2$ points, and 50 of 64 cells are at or below zero. No method exceeds the control by more than 1.2 points on any of the eight benchmarks, a ceiling reached once. The failures are not uniform: on AI2D every method loses 8 to 30 points. MMStar and ScienceQA losses are large but vary widely across methods, from $-1.8$ to $-39.1$ and from $-4.4$ to $-32.4$, respectively. MathVista shows a smaller, uniform loss (4 to 8 points across all eight). On POPE, MME, DocVQA, and LogicVista, differences are within noise of zero. Nowhere does selective machinery deliver a meaningful win over the simple random-refresh control.

**Cross-Family and Robustness Checks.** We next ask whether the primary-family pattern survives changes in model family, benchmark, and evaluation details. A complete factorial study would require adapting all eight methods, every evaluation harness and frame, and the seed protocol across every model family and benchmark; the resulting method $\times$ family $\times$ benchmark workload is beyond our available engineering time and compute. We therefore maximize coverage of these three dimensions under a fixed common protocol: the primary family receives the exhaustive eight-method $\times$ eight-benchmark evaluation, while the additional families and benchmarks are sampled to cover distinct task types and evaluation conditions. Across 34 additional judged cells on Gemma 4-E4B, Qwen3.5-4B, and Qwen3-VL-4B, 31 remain within the $+1.2$-point threshold. The three exceptions all occur on Qwen3.5-4B: SAPO exceeds random refresh by $+4.3$ points on AI2D and $+3.9$ on ChartQA, while Ent-80/20 reaches $+1.9$ on ScienceQA. We also concentrate robustness checks where they are most informative: identical seed budgets and strongest-valid-seed selection are used for every arm, and 12 additional same-frame seed checks preserve the original judgments. More detailed results are provided in Appendix A.

**Why Can Random Refresh Be Competitive?** The compared methods primarily modify how previously sampled rollouts contribute to optimization while leaving the underlying pool largely unchanged. Random refresh instead modifies the data itself, continuously replacing a small fraction of stored rollouts with samples generated closer to the current policy. A natural interpretation is that modest freshness can recover much of the benefit sought through more elaborate correction of stale data. Importantly, this does not imply that correction-based methods never help; rather, it shows that gains over a frozen pool alone are insufficient to establish an advantage over fresh data. Random refresh should therefore serve as a standard control whenever evaluating any newly proposed method for mitigating data staleness in practical deployments.

This result shifts the question from *how stale rollouts should be corrected* to *where staleness arises in the first place*. We investigate this next at the token level.

## 5 Finding 2: Staleness Tracks Visual Dependence

Finding 1 shows that modest random refresh can outperform substantially more elaborate treatments of stale data. This raises a more basic question: *where does staleness actually concentrate within a rollout?* We answer this at the token level by measuring policy drift and visual dependence.

**Measuring Token-Level Staleness and Visual Dependence.** For each token, we re-evaluate its log-probability under the trainee's current weights at every epoch and mark it as stale when the value falls below a fixed threshold ($-2.5$). We measure visual dependence as the size of the change

Table 2: Staleness-rate ratio of high- vs. low-$|\mathrm{VD}|$ tokens ($> 1$: visually dependent tokens go stale more often; –: undefined).

| | Gemma 4 | InternVL3 | Qwen3-VL | Qwen3.5 |
|---|---|---|---|---|
| AI2D | 14.4 | 3.8 | 27.2 | 10.3 |
| ChartQA | 21.7 | 3.3 | 30.0 | 7.6 |
| MathVista | 6.4 | 3.7 | 5.5 | 6.9 |
| MME | 2.5 | 3.8 | 4.2 | 4.0 |
| POPE | 5.4 | 3.8 | 4.5 | 4.5 |
| SciQA | 21.9 | – | 25.9 | 5.4 |
| MMStar | 2.7 | – | 2.9 | 3.0 |

Table 3: Explorer–trainee configurations (Section 6): five pairs from four model families by three vendors, with 1B–4B explorers and 4B–9B trainees; $K$ matches sampling budgets.

| Family | Explorer | Trainee | $K$ |
|---|---|---|---|
| Gemma | Gemma 4-E2B | Gemma 4-E4B | 12 |
| Qwen3.5 | Qwen3.5 4B | Qwen3.5 9B | 13 |
| Qwen3-VL | Qwen3-VL 4B | Qwen3-VL 8B | 12 |
| InternVL3-1B | InternVL3 1B | InternVL3 8B | 48 |
| InternVL3-2B | InternVL3 2B | InternVL3 8B | 24 |

in token log-probability when the input image is masked out entirely,

$$\mathrm{VD}_t = \log p(y_t \mid I, y_{<t}) - \log p(y_t \mid I_{\mathrm{blank}}, y_{<t}), \tag{1}$$

computed once at sampling time. We utilize $|\mathrm{VD}_t|$ because both positive and negative changes indicate sensitivity to visual input. Tokens above the fixed $|\mathrm{VD}|$ threshold (1.0) are referred to as high-$|\mathrm{VD}|$ tokens, while the remainder form the low-$|\mathrm{VD}|$ group. For each model–benchmark cell, we compare the staleness rates of these two groups at the final logged epoch. When a family–benchmark pair has several valid seeds, their ratios are averaged. This yields 26 valid cells out of 28 family–benchmark pairs. A dash (–) in Table 2 marks a ratio whose numerator or denominator is zero and is therefore not meaningful; in both such cells, no token reaches the $|\mathrm{VD}|$ threshold.

**Main Result.** Table 2 shows a remarkably consistent pattern. Across all 26 valid family–benchmark cells, high-$|\mathrm{VD}|$ tokens exhibit higher staleness rates than low-$|\mathrm{VD}|$ tokens. The mean staleness-rate ratio is 9.1, with a median of 4.9, and every valid cell lies above 1 ($p = 3.0 \times 10^{-8}$, two-sided sign test). In other words, tokens that depend more strongly on the visual input become stale substantially more often than tokens that are less visually dependent. Full measurement details, run-selection rules, and per-cell statistics for all four model families and 26 cells are provided in Appendix B, including the two undefined cells.

**Cross-Family Consistency.** The magnitude of the effect varies across architectures and benchmarks, but its direction is stable. Family-level mean ratios are 10.7 for Gemma 4-E4B, 3.7 for InternVL3-8B, 14.3 for Qwen3-VL-8B, and 6.0 for Qwen3.5-9B. InternVL3-8B shows the smallest and most uniform ratios, ranging from 3.3 to 3.8, whereas Gemma 4-E4B and Qwen3-VL-8B exhibit substantially larger benchmark-dependent gaps, reaching 21.9 and 30.0, respectively. The two undefined InternVL3-8B entries contain no tokens above the fixed $|\mathrm{VD}|$ threshold, so their ratios are left undefined rather than treated as zero. Thus, the concentration of staleness on visually dependent tokens is not driven by any single model family or by any one evaluation benchmark alone.

**Connection to Finding 1.** This pattern explains why the methods in Finding 1 rarely outperform random refresh. Most evaluated methods prioritize or correct samples using entropy, advantage magnitude, importance ratios, or clipping statistics, signals that do not directly capture visual dependence. VPPO and PGPO do incorporate visual signals, but use them to reweight optimization on already-sampled data rather than to decide which rollouts should be replaced. Existing mechanisms are therefore misaligned with the token-level axis along which staleness concentrates, whereas random refresh introduces fresh data without relying on any proxy signal. InternVL3-8B's narrow ratio band also makes it a robust primary family for Finding 1, independent of benchmark choice.

## 6 Method: Selective Refresh as a Feasibility Test

Finding 2 reveals a strong token-level association between visual dependence and staleness, but refresh decisions operate on entire rollout groups. We therefore ask whether this diagnostic can be translated into a practical partial-refresh strategy. Our goal is deliberately limited: to test the feasibility of selective refresh under a small refresh budget, rather than to assume the token-level diagnostic directly yields an optimal, ready-to-use policy at the level of entire rollout groups.

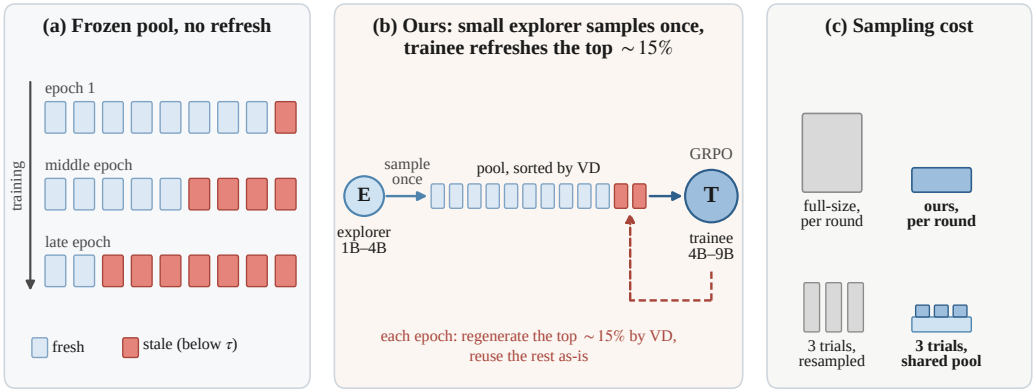


Figure 2: Overview of selective refresh. **(a)** Frozen-pool training repeatedly optimizes on the same rollouts as the policy evolves, causing the pool to become increasingly stale. **(b)** A small explorer samples the initial pool once; the larger trainee subsequently refreshes only the top $\sim$15% of groups according to a visual-dependence-derived score while reusing the remainder. **(c)** Partial refresh reduces full-size rollout generation and allows the initial pool to be reused across training runs.

**Selective Refresh.** As illustrated in Figure 2, we translate the token-level diagnostic from Finding 2 into a group-level refresh score derived from the same masked-image statistics. For each token, the explorer computes $s_t = -\log p_{\text{exp}}(y_t \mid I_{\text{blank}}, y_{<t})$, which equals $\text{VD}_t - \log p_{\text{exp}}(y_t \mid I, y_{<t})$ and therefore reuses the masked-image pass of Finding 2. Each response $r$ is summarized by $q_r = P_{90}(\{s_t\}_{t \in r})$ and $h_r = \sum_{t \in r} \mathbb{1}[s_t \geq 1]/|r|$, and each group $g$ is scored by its representative response,

$$S_b(g) = z(q_{r^*}) + z(h_{r^*}), \qquad r^* = \arg\max_{r \in g} q_r, \tag{2}$$

where $z(\cdot)$ standardizes a statistic across the pool. The top $\sim$15% of groups by $S_b$ are regenerated at each selective-refresh step, and the remaining pool is reused unchanged. The refresh fraction matches the random-refresh control, so the two strategies differ only in which groups are regenerated, not in how much fresh data they receive per epoch.

**Small Explorer, Large Trainee.** A small explorer (1B–4B) samples the initial rollout pool once, while a larger trainee (4B–9B) performs GRPO training. At each selective-refresh step, the trainee regenerates only the selected groups rather than the full pool. We evaluate five explorer–trainee configurations summarized in Table 3, with group size $K$ adjusted by configuration to keep sampling budgets comparable. This design reduces expensive full-size autoregressive generation while retaining the larger, more capable model as both the training target and the final policy.

**Performance against On-Policy GRPO.** Table 4 reports ten representative settings from three configurations; selective refresh exceeds on-policy GRPO in all ten, with margins ranging from $+0.8$ to $+8.4$ points. Eight of the ten also outperform random refresh; the two exceptions are InternVL3-1B$\rightarrow$8B on ChartQA and InternVL3-2B$\rightarrow$8B on MMStar, where random refresh remains slightly stronger. Across all five configurations, selective refresh achieves a positive margin over both random refresh and on-policy GRPO in 35 of 55 judged units (Table 7, Appendix C).

**Sampling Efficiency.** Selective refresh mainly cuts the compute of rollout generation. For 1B$\rightarrow$8B, the explorer is eight times smaller, so it can draw six times as many responses per prompt ($K = 48$ vs. 8) at 75% of the cost of a full-size trainee pool; with the 15% trainee refresh, a round costs about 90%, against 100% for the eight methods and 115% for random refresh (Figure 3a). Since the pool is sampled once and reused, eight epochs cost 22.5% of on-policy GRPO (13.1% from the trainee), versus 25.6% for random refresh (Figure 6). Adding the trainee forward over the larger pool at $0.1\times$ a generation gives a modelled sampling time of 117% of an on-policy round for 2B$\rightarrow$8B, close to random refresh (115%); the gap to the frozen-pool methods (100%) is the cost of refreshing (Figure 3b). Measured end-to-end wall-clock for the pure selective-refresh arm is 20–96% of on-policy GRPO across all timed configurations and benchmarks (Table 10). These numbers characterize sampling efficiency rather than total training FLOPs (sampling dominates cost, but optimization runs over a larger pool of responses per group).

Table 4: Selective refresh versus on-policy GRPO on ten model–benchmark settings. Our method reaches or exceeds on-policy GRPO in all 10 settings. $\Delta_{\mathrm{OP}}$ reports the accuracy margin over on-policy GRPO in percentage points; random refresh is shown for reference.

| Family | Benchmark | Random Refresh | On-Policy GRPO | Ours | $\Delta_{\mathrm{OP}}$ |
|---|---|---|---|---|---|
| InternVL3-1B→8B | HallusionBench | 0.672 | 0.601 | **0.685** | +8.4 |
| | ChartQA | **0.589** | 0.534 | 0.567 | +3.3 |
| | AI2D | 0.794 | 0.802 | **0.819** | +1.7 |
| | MMStar | 0.573 | 0.597 | **0.607** | +1.0 |
| InternVL3-2B→8B | HallusionBench | 0.601 | 0.601 | **0.652** | +5.1 |
| | MMStar | **0.630** | 0.597 | 0.625 | +2.8 |
| | MMMU | 0.486 | 0.479 | **0.503** | +2.4 |
| | ChartQA | 0.524 | 0.527 | **0.535** | +0.8 |
| Qwen3.5-4B→9B | BLINK | 0.579 | 0.566 | **0.590** | +2.4 |
| | AI2D | 0.810 | 0.829 | **0.869** | +4.0 |

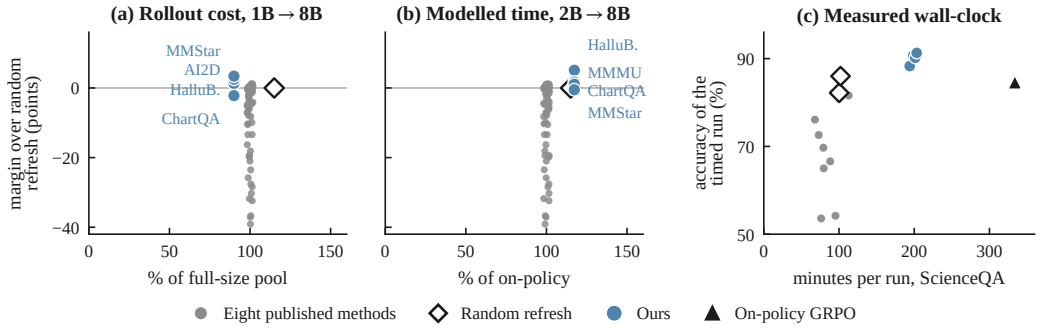


Figure 3: Sampling cost and measured wall-clock of selective refresh. **(a)** Rollout cost per round for 1B→8B, as a share of a full-size pool (8 responses per prompt), and **(b)** modelled sampling time per round for 2B→8B, which adds the trainee forward at $0.1\times$ a generation, as a share of on-policy GRPO; both are shown against accuracy margin over random refresh. **(c)** Measured end-to-end wall-clock and final accuracy on ScienceQA, with one point per timed run.

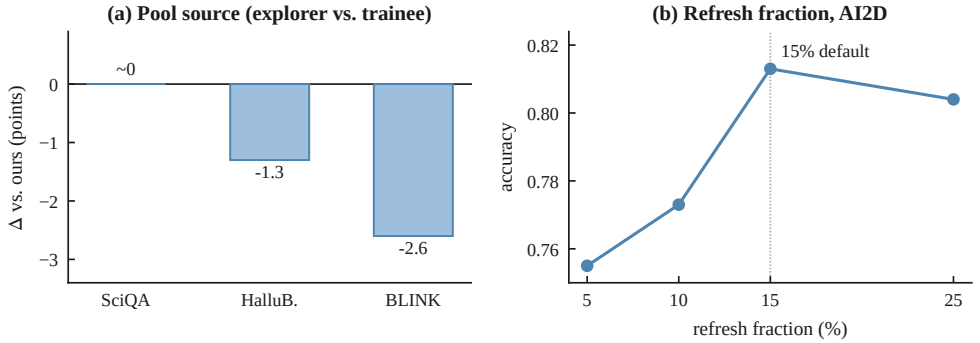


Figure 4: Ablations isolating where the accuracy gains come from. **(a)** Changing the pool source at matched size does not outperform our method across three same-frame benchmarks. **(b)** Accuracy improves from 5% to the 15% default refresh rate, while increasing refresh further provides no additional gain.

## 7 Where the Gains Come From

Section 6 shows that partial refresh matches or beats full on-policy resampling. We next ask what actually drives this gain. We examine four possibilities: the initial rollout pool, the amount of refresh, the choice of which groups to regenerate, and the precision of the refresh signal.

**Is the Gain Explained by the Initial Pool?** We first test whether the improvement comes from the explorer-sampled pool rather than refresh itself. Holding pool size fixed, replacing the explorer-

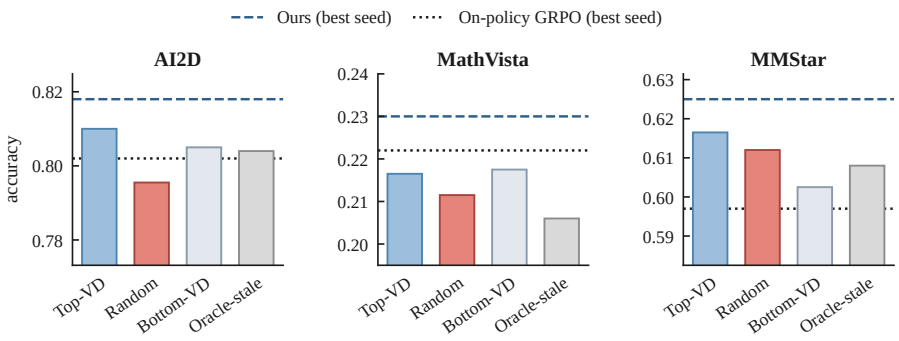


Figure 5: Equal-budget refresh-target intervention for InternVL3-2B→8B. Holding the pool, trainee, and $\sim$15% refresh budget fixed, we compare Top-VD, random, Bottom-VD, and Oracle-stale targeting. Bars show seed means; horizontal lines mark the best selective-refresh and on-policy GRPO results that were obtained on these exact same evaluation cells.

sampled pool with a trainee-sampled one does not outperform our method on ScienceQA, HallusionBench, or BLINK, with margins of $0.0$, $-1.3$, and $-2.6$ points (Figure 4a). Doubling the pool size from $K = 48$ to $K = 96$ on AI2D likewise reduces accuracy by $0.6$ points. The random-refresh control itself is sensitive to pool source in a venue-dependent way: on HallusionBench, an explorer-sampled pool beats a trainee-sampled one by $13.6$ points ($p < 10^{-4}$), with no consistent advantage elsewhere. Pool source and pool size alone therefore cannot explain the gain of selective refresh over both controls, since every arm here shares the same trainee and training budget.

**How Much Refresh Is Needed?** Under the same refresh schedule, AI2D accuracy increases from $0.755$ at 5% refresh to $0.773$ at 10% and $0.813$ at the 15% default (Figure 4b). Increasing the fraction to 25% yields $0.804$. On Qwen3-VL-4B→8B AI2D, 25% and 35% refresh both average $0.786$, below the best default-rate result of $0.812$. The evidence nevertheless shows that modest refresh is important, though increasing the fraction further does not guarantee improvement.

**Does the Refresh Target Matter?** We isolate the targeting rule directly by fixing the initial pool, trainee, training budget, and $\sim$15% refresh fraction, and varying only which groups are regenerated. Figure 5 compares four strategies: Top-VD, uniform random refresh, Bottom-VD, and Oracle-stale, which targets groups using measured staleness directly. Top-VD, which ranks groups purely by visual dependence, is the most consistent across the three benchmarks, supporting the VD-derived score $S_b$ of Section 6 as the default selection criterion in all our main experiments.

## 8 Conclusion

We revisited how staleness should be handled when rollouts are reused across epochs in GRPO-style post-training of VLMs, where rollout generation dominates the training cost. Under a common protocol, eight methods that can be used to mitigate data staleness rarely improve on refreshing a small random fraction of the rollout pool, both on the primary model family and in spot checks on three further families. Measuring staleness during training shows where it concentrates: visually dependent tokens go stale more often than language-prior tokens in all 26 cells across the four model families we examined. Building on these findings, we proposed a selective-refresh method that turns this token-level diagnostic into a group-level refresh rule; its results are consistent with both findings, and it mitigates staleness at a lower sampling cost than full on-policy resampling. Together, these results suggest that random refresh should serve as a standard control when evaluating methods for handling stale rollouts in GRPO-style post-training.

**Limitations.** Our compute resources are limited, so rather than concentrating them on a single configuration, we aimed to cover as broad a range as possible within this budget; this is why we evaluate one model family exhaustively and cover the other families and benchmarks through spot checks. For the same reason, the training data scale of each run is limited; we expect both findings and the advantage of selective refresh to become more pronounced at a larger training scale, which our current compute resources do not allow us to test.

### REPRODUCIBILITY STATEMENT

We will release all training and evaluation code, run logs, validity criteria, significance scripts, and audit records required to reproduce the reported results. All reported numbers are traceable to logged runs under the stated protocol, including any judgment changes identified during auditing. Dataset- and benchmark-specific reproducibility details are provided in the appendix.

### AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and editing, literature review, formatting, and the interpretation and presentation of existing results. The authors are responsible for verifying all AI-assisted content and take full responsibility for the final manuscript.

## REFERENCES

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *NeurIPS*, 2024.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. MME: A comprehensive evaluation benchmark for multimodal large language models. In *NeurIPS*, 2025.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. BLINK: Multimodal large language models can see but not perceive. In *ECCV*, pp. 148–166, 2024.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

Gemma Team. Gemma 4 technical report. *CoRR*, abs/2607.02770, 2026.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. HallusionBench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *CVPR*, pp. 14375–14385, 2024.

Siyuan Huang, Xiaoye Qu, Yafu Li, Yun Luo, Zefeng He, Daizong Liu, and Yu Cheng. Spotlight on token perception for multimodal reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 103291–103325, 2026.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *ECCV*, pp. 235–251, 2016.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *EMNLP*, pp. 292–305, 2023.

Yunheng Li, Hangyi Kuang, Hengrui Zhang, Jiangxia Cao, Zhaojie Liu, Qibin Hou, and Ming-Ming Cheng. Rethinking token-level policy optimization for multimodal chain-of-thought. *arXiv preprint arXiv:2603.22847*, 2026.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *ICLR*, 2024.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *WACV*, pp. 2199–2208, 2021.

Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. Asynchronous rlhf: Faster and more efficient off-policy rl for language models. In *International Conference on Learning Representations*, volume 2025, pp. 4003–4029, 2025.

Zekai Qu, Yinxu Pan, Ao Sun, Chaojun Xiao, and Xu Han. Copris: Efficient and stable reinforcement learning via concurrency-controlled partial rollout with importance sampling. *arXiv preprint arXiv:2511.05589*, 2025.

Qwen Team. Qwen3-vl technical report. *CoRR*, abs/2511.21631, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Yiming Ren, Yiran Xu, Zicheng Lin, Chufan Shi, Yukang Chen, Dingdong Wang, Tianhe Wu, Junjie Wang, Yujiu Yang, Yu Qiao, and Ruihang Chu. Smaller models are natural explorers for policy-level diversity in GRPO. *arXiv preprint arXiv:2605.30789*, 2026.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

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.

Minghao Tian, Yunfei Xie, and Chen Wei. How off-policy can grpo be? mu-grpo for efficient llm reinforcement learning. *arXiv preprint arXiv:2605.17570*, 2026.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:115452–115486, 2026.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Xun Deng, Zhihao Zhang, et al. Bapo: Stabilizing off-policy reinforcement learning for llms via balanced policy optimization with adaptive clipping. In *International Conference on Learning Representations*, volume 2026, pp. 126204–126228, 2026.

Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. LogicVista: Multimodal LLM logical reasoning benchmark in visual contexts. *CoRR*, abs/2407.04973, 2024.

Yixuan Even Xu, Yash Savani, Fei Fang, and J Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

Zekai Ye, Qiming Li, Xiaocheng Feng, Ruihan Chen, Ziming Li, Haoyu Ren, Kun Chen, Dandan Tu, and Bing Qin. Not all tokens see equally: Perception-grounded policy optimization for large vision-language models. *arXiv preprint arXiv:2604.01840*, 2026.

Gyeongtae Yoo, Sanghyeok Park, Soohyuk Jang, Ik-hwan Kim, and Sungroh Yoon. Rollout-level advantage-prioritized experience replay for grpo. *arXiv preprint arXiv:2606.04560*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026a.

Song Yu, Li Li, Wenwen Zhao, and Zhisheng Yang. Erpo: Token-level entropy-regulated policy optimization for large reasoning models. *arXiv preprint arXiv:2603.28204*, 2026b.

Xiang Yue, Yuansheng Ni, Tianyu Zheng, Kai Zhang, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, pp. 9556–9567, 2024.

Haizhong Zheng, Jiawei Zhao, and Beidi Chen. Prosperity before collapse: How far can off-policy rl reach with stale data on llms? In *International Conference on Learning Representations*, volume 2026, pp. 82657–82679, 2026.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *CoRR*, abs/2504.10479, 2025.

# APPENDIX

## APPENDIX ROADMAP

The appendix follows the paper's three empirical questions in order. Appendix A gives the complete method–benchmark grid and robustness checks for Finding 1; Appendix B gives the token-level staleness measurements for Finding 2; Appendix C gives the selective-refresh protocol, full results, and ablations; Appendix D records the compute and wall-clock accounting, and Appendix E gives training, scoring, and terminology details. Evaluation-frame caveats and seed-level audit notes are kept with the results they qualify.

## A EXTENDED RESULTS FOR FINDING 1

This section provides the complete primary-family results and cross-family spot checks supporting Finding 1. Robustness analyses, including seed stability, are reported separately in the following subsection.

### A.1 FULL AND CROSS-FAMILY RESULTS

**Primary-Family Grid.** Table 5 reports the eight-benchmark main-text evaluation on InternVL3-8B together with the random-refresh reference accuracies. All 64 method–benchmark cells remain at or below the $+1.2$-point threshold.

Table 5: Complete Finding 1 results on InternVL3-8B. The first row gives the random-refresh reference accuracy, while all remaining entries report accuracy margins over that reference in percentage points.

| Method | MMStar$^{\dagger}$ | POPE$^{\dagger}$ | MME$^{\dagger}$ | MathV.$^{\dagger}$ | SciQA | DocVQA | LogicV. | AI2D$^{\ddagger}$ |
|---|---|---|---|---|---|---|---|---|
| RR reference | 0.548 | 0.917 | 0.870 | 0.258 | 0.860 | 0.448 | 0.409 | 0.811 |
| VPPO | −36.7 | −2.4 | −4.8 | −5.2 | −9.9 | 0.0 | −0.2 | −30.3 |
| PGPO | −37.1 | +1.1 | −4.5 | −7.0 | −13.4 | −0.1 | −1.5 | −25.8 |
| Ent-80/20 | −28.4 | +0.3 | −2.9 | −5.0 | −16.3 | −0.5 | +0.7 | −27.6 |
| M2PO | −23.5 | −0.2 | +0.5 | −4.1 | −4.4 | −0.4 | −1.1 | −19.4 |
| Mu-GRPO | −13.4 | +0.6 | +0.1 | −5.8 | −21.0 | +0.1 | −1.1 | −19.8 |
| BAPO | −39.1 | +0.1 | +0.9 | −6.1 | −31.8 | −0.4 | −2.0 | −8.2 |
| Rollout-AP | −10.5 | +0.1 | +1.2 | −7.9 | −32.4 | −0.4 | −3.7 | −10.3 |
| SAPO | −1.8 | +1.1 | +0.2 | −7.2 | −19.4 | +0.3 | −0.2 | −18.1 |

**Cross-Family Spot Checks.** We further test Finding 1 on Gemma 4-E4B, Qwen3.5-4B, and Qwen3-VL-4B. As shown in Table 6, 31 of 34 additional judged cells remain within the $+1.2$-point threshold. All three exceptions occur on Qwen3.5-4B. SAPO exceeds random refresh by $+4.3$ points on AI2D and $+3.9$ points on ChartQA, while Ent-80/20 reaches $+1.9$ points on ScienceQA. Thus, the primary-family pattern largely transfers across the additional families, while retaining a small number of genuine exceptions.

**Exceptions and Evaluation-Frame Notes.** Across the 64 primary-family cells and 34 cross-family spot checks, three cells exceed the $+1.2$-point threshold, all of them the Qwen3.5-4B exceptions in Table 6. Rollout-AP on MME reaches exactly $+1.2$ points and therefore remains within the stated threshold.

In Table 5, $^{\dagger}$ denotes carved benchmarks for which the method runs and random-refresh references use different evaluation frames. These margins are retained with this caveat, while a strict same-frame re-check on MathVista preserves all eight judgments. For MME, the random-refresh reference is the mean of five clean seeds, yielding an accuracy of 0.870. The $^{\ddagger}$ marker indicates that both AI2D random-refresh seeds finish below the initial accuracy $a_0 = 0.875$, making the reference weaker rather than stronger.

Table 6: Cross-family spot checks for Finding 1. Each row reports the same-frame random-refresh reference and method accuracies, with margins over the reference in parentheses. The final column gives the number of evaluated methods remaining within the $+1.2$-point threshold.

| Family | Venue (slice, $a_0$) | Random-refresh reference and method results | Within $+1.2$ |
|---|---|---|---|
| Gemma 4-E4B | ChartQA (2493, 0.800) | Ref. 0.728 (0.728, 0.722); M2PO 0.722 ($-0.6$), Rollout-AP 0.712 ($-1.6$), BAPO 0.711 ($-1.7$) | 3/3 |
| Qwen3.5-4B | ChartQA (2493, 0.800) | Ref. 0.760 (0.760, 0.756); PGPO 0.761 ($+0.1$), SAPO 0.799 ($+\mathbf{3.9}$) | 1/2 |
| Qwen3-VL-4B | ChartQA (2493, 0.800) | Ref. 0.750 (0.750, 0.742); SAPO 0.757 ($+0.7$), Ent-80/20 0.760 ($+1.0$), Rollout-AP 0.750 (0.0; seeds 0.750, 0.748), PGPO 0.752 ($+0.2$) | 4/4 |
| Gemma 4-E4B | ScienceQA (1921, 0.675) | Ref. 0.814 (0.814, 0.806, 0.811); BAPO 0.811 ($-0.3$), Ent-80/20 0.805 ($-0.9$), M2PO 0.817 ($+0.3$, $n$=2), PGPO 0.812 ($-0.2$), VPPO 0.814 (0.0), Mu-GRPO 0.816 ($+0.2$), Rollout-AP 0.824 ($+1.0$) | 7/7 |
| Qwen3.5-4B | ScienceQA (1921, 0.825) | Ref. 0.893 (0.893, 0.873, 0.888); Ent-80/20 0.912 ($+\mathbf{1.9}$), Mu-GRPO 0.901 ($+0.8$), SAPO 0.902 ($+0.9$), Rollout-AP 0.898 ($+0.5$), VPPO 0.865 ($-2.8$), M2PO 0.901 ($+0.8$) | 5/6 |
| Qwen3.5-4B | AI2D (3022, 0.900) | Ref. 0.808 (0.776, 0.808); BAPO 0.799 ($-0.9$), M2PO 0.798 ($-1.0$), PGPO 0.744 ($-6.4$), SAPO 0.851 ($+\mathbf{4.3}$) | 3/4 |
| Gemma 4-E4B | AI2D (3022, 0.775) | Ref. 0.768 (0.768, 0.767); BAPO 0.765 ($-0.3$), Ent-80/20 0.774 ($+0.6$), M2PO 0.772 ($+0.4$), Mu-GRPO 0.770 ($+0.2$), PGPO 0.768 (0.0), Rollout-AP 0.772 ($+0.4$), SAPO 0.780 ($+1.2$), VPPO 0.778 ($+1.0$) | 8/8 |

Several cross-family random-refresh references on ChartQA and AI2D also finish below their corresponding initial accuracies. These margins should therefore be interpreted with the corresponding reference-degradation caveat.

### A.2 Robustness Checks

We further examine whether Finding 1 is sensitive to seed variation.

**Seed Stability.** We rerun SAPO, M2PO, and Rollout-AP with two additional seeds on selected benchmarks using the same evaluation frames as the original comparisons. All 12 valid same-frame checks on DocVQA and ScienceQA preserve the original judgments. The additional MMStar runs fall on a different evaluation frame and are therefore excluded from this check. Although absolute accuracies can vary across seeds, the same-frame comparisons remain stable under the matched seed protocol. For example, incorporating the additional valid ScienceQA seeds changes the margins of M2PO and Rollout-AP to $-1.8$ and $-2.9$ points, respectively, without changing either conclusion. These audit seeds were run only to test reproducibility and, by design, are not added to the strongest-seed pool behind Tables 1 and 5, whose entries therefore remain unchanged.

## B Additional Analysis for Finding 2

This section provides additional measurement details and statistics for Finding 2. All reported results use the magnitude-based visual-dependence split described below.

### B.1 Measurement Protocol

**Token-Level Staleness and Visual Dependence.** The statistics in Figure 1(b) and Table 2 are collected directly during training rather than by a separate post-hoc diagnostic. At every frozen-pool epoch, each token is re-evaluated under the trainee's current weights after the policy has undergone additional gradient updates. A token is marked stale when its log-probability falls below the fixed threshold of $-2.5$. Visual dependence is measured using $|\mathrm{VD}|$ as defined in Section 5, with a fixed threshold $\tau_{\mathrm{VD}} = 1.0$ separating high-$|\mathrm{VD}|$ and low-$|\mathrm{VD}|$ tokens.

For each cell, we compute the two group-specific staleness rates

$$r_{\mathrm{hi}} = \frac{\#\{t : \mathrm{stale}_t,\ |\mathrm{VD}_t| \geq \tau_{\mathrm{VD}}\}}{\#\{t : |\mathrm{VD}_t| \geq \tau_{\mathrm{VD}}\}}, \tag{3}$$

and

$$r_{\mathrm{lo}} = \frac{\#\{t : \mathrm{stale}_t,\ |\mathrm{VD}_t| < \tau_{\mathrm{VD}}\}}{\#\{t : |\mathrm{VD}_t| < \tau_{\mathrm{VD}}\}}. \tag{4}$$

The reported statistic is the staleness-rate ratio

$$R_{\mathrm{VD}} = \frac{r_{\mathrm{hi}}}{r_{\mathrm{lo}}}, \tag{5}$$

where $R_{\mathrm{VD}} > 1$ indicates that visually dependent tokens become stale more frequently than low-$|\mathrm{VD}|$ tokens.

**Run Selection and Aggregation.** We restrict this analysis to the self-sampled frozen-pool regime used by Findings 1–2. Runs involving selective refresh or different explorer and trainee models are excluded so that the measured drift arises only from within-training policy evolution. For each qualifying run, we use the final logged epoch for which both high-$|\mathrm{VD}|$ and low-$|\mathrm{VD}|$ staleness rates are defined and nonzero. When multiple valid seeds exist for the same family–benchmark pair, their ratios are averaged before entering the cross-cell statistics, so each family–benchmark pair contributes exactly once. For example, the two Qwen3-VL×AI2D seeds yield ratios of 24.6 and 29.9, producing the 27.2 value reported in Table 2. This procedure yields 26 valid cells out of 28 possible family–benchmark pairs.

### B.2 Additional Statistics

**Per-Family Statistics.** All 26 valid cells satisfy $R_{\mathrm{VD}} > 1$, with a mean ratio of 9.1, a median of 4.9, and a two-sided sign-test value of $p = 3.0 \times 10^{-8}$. Gemma 4-E4B contributes seven cells with a mean ratio of 10.7, while InternVL3-8B contributes five cells with a mean of 3.7. Qwen3-VL-8B contributes seven cells with the largest family-level mean of 14.3, and Qwen3.5-9B contributes seven cells with a mean of 6.0. Thus, the magnitude varies substantially across model families, while the direction of the effect remains unchanged.

**Undefined Cells.** The two undefined cells are InternVL3-8B on ScienceQA and MMStar. Neither run contains tokens reaching $|\mathrm{VD}| \geq 1.0$, so the high-$|\mathrm{VD}|$ staleness rate and its ratio are undefined. We therefore exclude these cells rather than assigning a zero ratio.

## C Selective Refresh: Full Results and Analysis

This section provides the implementation details, complete evaluation protocol, and additional analyses for the selective-refresh feasibility study in Sections 6–7. We first specify the group-level refresh score and training schedules, followed by the validity and judging rules used throughout the full evaluation.

### C.1 Implementation and Evaluation Protocol

**Group-Level Refresh Score.** We derive the group-level refresh score from the same masked-image statistics used to measure visual dependence in Finding 2. For each token $t$, we define

$$s_t = \mathrm{VD}_t - \log p_{\mathrm{exp}}(y_t \mid I, y_{<t}). \tag{6}$$

Using

$$\mathrm{VD}_t = \log p_{\mathrm{exp}}(y_t \mid I, y_{<t}) - \log p_{\mathrm{exp}}(y_t \mid I_{\mathrm{blank}}, y_{<t}), \tag{7}$$

the score simplifies to

$$s_t = -\log p_{\mathrm{exp}}(y_t \mid I_{\mathrm{blank}}, y_{<t}). \tag{8}$$

Thus, the score is obtained from the same real-image and masked-image forward passes already used for visual-dependence estimation, without an additional trainee-side forward pass.

For each response $r$, we compute the 90th percentile of its token scores,

$$q_r = P_{90}\left(\{s_t\}_{t \in r}\right). \tag{9}$$

We also compute the fraction of tokens whose scores reach the fixed cutoff of 1.0,

$$h_r = \frac{1}{|r|} \sum_{t \in r} \mathbb{1}[s_t \geq 1.0]. \tag{10}$$

Within each rollout group $g$, the response with the largest $q_r$ is selected as the representative response. Its $q_r$ and $h_r$ values are independently standardized across the rollout pool and summed to obtain the group score $S_b$. At each selective-refresh step, the top $\sim 15\%$ of rollout groups ranked by $S_b$ are regenerated, while the remaining groups are reused unchanged.

**Refresh Schedules.** We evaluate three selective-refresh schedules that share the same group score and nominal $\sim 15\%$ refresh fraction. *Always* (A) applies selective refresh throughout training. *Early* (E) applies selective refresh only during the early stage of training. *Early-to-On-Policy* (EO) begins with selective refresh and later switches to full on-policy resampling. All three schedules use the same initial explorer-sampled pool and differ only in when selective refresh is active.

**Validity and Judging Rules.** Each evaluation unit uses 200 rollout groups and eight training epochs, with the group size $K$ determined by the explorer–trainee configuration in Table 3. The on-policy reference instead re-samples every rollout group at each epoch with $K = 6$. A run is considered valid only if evaluation completes without OOM, the optimizer executes at least one step, the exception count does not exceed 10% of optimization steps, and the trained adapter matches the expected family-specific tensor count. The expected adapter tensor counts are 160 for Qwen3.5, 224 for InternVL3, 264 for Gemma, and 288 for Qwen3-VL.

Comparisons are judged only within matched evaluation frames. A frame requires the same evaluation slice and the same initial accuracy $a_0$ across the compared arms. When two valid evaluation frames exist for the same model–benchmark pair, each frame is treated as a separate evaluation unit. InfoVQA and TextVQA are excluded from judging because their evaluation pipelines are flagged.

The initial accuracy $a_0$ is measured on a fixed 40-item subset before training, while the final accuracies used for all reported margins are measured on the complete evaluation slice. Because $a_0$ is estimated from a smaller subset, it is used to identify matched frames rather than as the primary performance statistic.

A unit is counted as a positive selective-refresh result only when the strongest valid seed among the three schedules exceeds both the strongest random-refresh seed and the strongest on-policy seed under identical seed budgets. For each unit, we report the more conservative of the two margins,

$$\Delta_{\text{both}} = \min\left(A_{\text{ours}} - A_{\text{RR}}, A_{\text{ours}} - A_{\text{OP}}\right), \tag{11}$$

where $A_{\text{RR}}$ and $A_{\text{OP}}$ denote the matched random-refresh and on-policy references, respectively. We additionally examine seed means when sufficient seeds are available, while the complete seed-level record and statistical tests are reported below.

### C.2 Complete Results and Statistical Analysis

**Full Five-Configuration Grid.** Table 7 reports the complete selective-refresh evaluation across five explorer–trainee configurations. For each judged unit, we report

$$\Delta_{\text{both}} = \min\left(A_{\text{ours}} - A_{\text{RR}}, A_{\text{ours}} - A_{\text{OP}}\right), \tag{12}$$

so a positive value indicates that selective refresh exceeds both random refresh and on-policy GRPO. Across the complete evaluation, 35 of 55 judged units have positive margins over both controls. The remaining units are retained in the denominator, including ties and negative margins.

Table markers denote reference or evaluation caveats. Superscript $g$ indicates that the random-refresh reference finishes below its initial accuracy. Superscript $b$ indicates that the frozen-pool baseline ties or exceeds the strongest selective-refresh result. Superscript $f$ denotes model–benchmark pairs with two valid evaluation frames that are judged separately. Superscript $o$ denotes units excluded because the on-policy reference fails the validity criteria. Superscript $c$ denotes a configuration for which no canonical evaluation frame is available.

**Seed-Level Results.** The aggregate result above is based on the strongest valid seed under identical seed budgets for selective refresh and both controls. Most positive margins are modest, which motivates treating the 35-of-55 result as a directional aggregate rather than a statistical-significance claim. Three units remain positive under all three selective-refresh schedules: InternVL3-2B→8B on MMMU, Gemma-2B→4B on POPE, and InternVL3-2B→8B on the local ChartQA frame. These observations motivate the explicit significance tests below.

Table 7: Complete selective-refresh evaluation across five explorer–trainee configurations. Entries report $\Delta_{\text{both}}$ in percentage points, defined as the smaller accuracy margin over random refresh and on-policy GRPO. Positive values therefore indicate improvement over both controls.

| Family | AI2D | BLINK | HalluB. | MMMU | MME | MathV. | POPE | SciQA | ChartQA | MMStar | DocVQA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Gemma 4 (E2B→E4B) | n/a[c] | +0.5 | +0.4[b] | +0.1 | +0.1 | +1.1[g] | +0.4[b] | +0.4 | −0.4 | +0.4 | ±0.0 |
| Qwen3.5 (4B→9B) | +4.0[b] | +1.1 | +0.2 | +0.4 | −1.2 | +0.5[g] | −2.2 | +0.1 | −0.1 | +0.5[g] | −0.2 |
| Qwen3-VL (4B→8B) | +0.5[gb] | +0.7 | +0.6[g] | +0.6 | +0.5[b] | −0.7 | +1.4 | +0.6[b] | +0.2 | n/a[o] | −0.3 |
| InternVL3 (1B→8B) | +1.7[g] | +0.7 | +1.3 | ±0.0 | −0.8 | +0.8[g] | −5.3 | −0.1 | −2.2 | +1.0[g] | −0.7 |
| InternVL3 (2B→8B) | +0.7 | −0.5 | +5.1[g] | +1.7 | +0.4 | +0.8[g] | −3.0 | +0.1 / − 0.4[f] | +0.8 / − 1.3[f] | −0.5[g] | −0.7 |

Table 8: Statistical tests for comparisons with sufficient valid seeds on the current evaluation frames. We report the mean accuracy difference, two-sided Welch's $t$-test, Mann–Whitney $U$, and the 95% confidence interval on the mean difference.

| Cell | Comparison | $\Delta$ mean | Welch $p$ | MWU $p$ | 95% CI | Result |
|---|---|---|---|---|---|---|
| HalluB., pool source | 1B-explorer RR (5) vs. 8B self RR (3) | +0.1357 | $< 0.0001$ | 0.0357 | [+0.108, +0.163] | Sig. |
| Gemma BLINK | A (6) vs. OP (4) | +0.0573 | 0.0050 | 0.0095 | [+0.029, +0.086] | Sig. |
| Gemma BLINK | E (6) vs. OP (4) | +0.0535 | 0.0042 | 0.0190 | [+0.023, +0.084] | Sig. |
| Gemma BLINK | A (6) vs. RR (8) | +0.0048 | 0.413 | 0.345 | [−0.008, +0.017] | n.s. |
| Gemma BLINK | E (6) vs. RR (8) | +0.0010 | 0.925 | 0.755 | [−0.023, +0.025] | n.s. |
| Qwen3.5 HalluB. | EO (11) vs. OP (7) | +0.0217 | 0.0023 | 0.0041 | [+0.009, +0.034] | Sig. |
| Qwen3.5 HalluB. | A (8) vs. RR (17) | +0.0065 | 0.451 | 0.511 | [−0.011, +0.024] | n.s. |
| Qwen3.5 HalluB. | E (7) vs. RR (17) | −0.0008 | 0.921 | 0.710 | [−0.017, +0.016] | n.s. |
| Qwen3.5 HalluB. | EO (11) vs. RR (17) | +0.0195 | 0.0104 | 0.0326 | [+0.005, +0.034] | Sig. |
| Qwen3.5 HalluB. | A (8) vs. OP (7) | +0.0087 | 0.274 | 0.397 | [−0.008, +0.025] | n.s. |
| Qwen3.5 HalluB. | E (7) vs. OP (7) | +0.0014 | 0.839 | 1.000 | [−0.014, +0.017] | n.s. |

**Statistical Significance.** We perform two-sided Welch's $t$-tests and Mann–Whitney $U$ tests only for comparisons with sufficient valid seeds. Table 8 reports tests that remain associated with the current evaluation frames.

The strongest direct evidence occurs on Qwen3.5-4B→9B HallusionBench. The EO schedule exceeds on-policy GRPO by 2.17 points on seed means and random refresh by 1.95 points, with both comparisons significant under Welch's $t$-test and Mann–Whitney $U$. On Gemma-2B→4B BLINK, the A and E schedules significantly exceed on-policy GRPO but not random refresh. The pool-source intervention on HallusionBench also shows a significant difference between explorer-sampled and self-sampled random-refresh pools. The $p < 10^{-4}$ quoted for this comparison in Section 7 is the Welch's $t$-test value; with five versus three seeds, the smallest attainable two-sided Mann–Whitney $p$ is $0.0357$, which is the value reported in Table 8. Overall, these tests reinforce that the full-grid result should primarily be interpreted as a margin-based feasibility result rather than a uniformly significant improvement across cells.

### C.3 Additional Ablations and Audit Notes

**Pool-Source and Pool-Size Controls.** We first test whether the gain can be explained by the explorer-sampled initial pool rather than selective refresh itself. At matched pool size, replacing the explorer-sampled pool with a trainee-sampled pool does not outperform our method on ScienceQA, HallusionBench, or BLINK, with margins of approximately $0.0$, $-1.3$, and $-2.6$ points relative to our method, respectively. Doubling the pool size from $K = 48$ to $K = 96$ on AI2D likewise reduces accuracy by $0.6$ points. The random-refresh control itself also shows venue-dependent sensitivity to pool source, with no consistent advantage for either explorer- or trainee-sampled pools. These results indicate that neither pool source nor pool size alone explains the observed selective-refresh gains.

**Refresh Fraction.** We further examine whether performance improves monotonically with the amount of refresh. Under the same plain refresh schedule on InternVL3-1B→8B AI2D, accuracy increases from $0.755$ at 5% refresh to $0.773$ at 10% and $0.813$ at the 15% default. The 25% setting reaches $0.804$, but uses the early-refresh schedule and therefore does not form a strict single-schedule sweep with the lower rates. On Qwen3-VL-4B→8B AI2D, the 25% and 35% settings both average $0.786$, compared with a best default-rate result of $0.812$. Thus, increasing the refresh fraction does not guarantee better downstream performance.

**Refresh-Target Intervention.** We directly isolate the choice of refresh target using the equal-budget intervention in Figure 5. All four strategies share the same InternVL3-2B→8B configuration, initial pool, 15% refresh fraction, and eight-epoch training budget, differing only in which groups are regenerated. For Top-VD, Random, Bottom-VD, and Oracle-stale, the seed-mean accuracies are $0.810$, $0.796$, $0.805$, and $0.804$ on AI2D; $0.217$, $0.212$, $0.218$, and $0.206$ on MathVista; and $0.617$, $0.612$, $0.603$, and $0.608$ on MMStar. These intervention runs are excluded from the 55-unit aggregate in Table 7.

**Statistical Scope and Audit Trail.** The full evaluation prioritizes breadth across model families and benchmarks rather than deep replication of every individual cell. Accordingly, the 35-of-55 result in Table 7 should be interpreted primarily as a margin-based feasibility result rather than a uniform significance claim. Among comparisons on the current evaluation frames with sufficient seeds, Qwen3.5-4B→9B on HallusionBench provides the strongest evidence, with the EO schedule significantly exceeding both random refresh and on-policy GRPO under both statistical tests in Table 8. As additional valid seeds were incorporated, 13 unit-level judgments changed direction, with six moving from positive to non-positive and seven moving in the opposite direction, producing a net change of one positive unit. The released audit record preserves these judgment changes and their causes.

# D Cost Accounting

This section details the sampling-cost model and measured wall-clock results underlying Figure 3. We distinguish rollout-generation cost, modelled sampling time, and end-to-end wall-clock, and do not interpret any of them as total training FLOPs.

## D.1 Cost Model and Per-Round Accounting

**Notation and Cost Model.** We normalize one full-size rollout pool to 100 generation units: $N = 200$ prompts with eight trainee responses each, the pool size used by the eight reproduced methods and by random refresh; on-policy GRPO is compared at this same per-round pool size, although the on-policy runs themselves sample six responses per prompt (75 units per round). A round consists of building the pool once plus one refresh. Each refresh regenerates a nominal fraction $\rho_r = 0.15$ of a full-size pool with the trainee, i.e., 15 units (as run, 30 groups with six trainee responses each). Let

$$\rho = \frac{P_e}{P_t} \tag{13}$$

denote the explorer-to-trainee parameter ratio. We approximate the generation cost of an explorer response as $\rho$ times that of a trainee response. The explorer pool uses $K_e = 6/\rho$ responses per prompt (Table 3; $K_e = 48$ for InternVL3-1B→8B and 24 for InternVL3-2B→8B), so at the nominal parameter ratios it costs $100\, K_e\rho/8 \approx 75$ units in every configuration.

Sampling also requires a teacher-forced trainee forward over every sampled response to obtain the log-probabilities used by GRPO. We denote its cost, relative to one trainee generation, by $F$. Because the explorer pool contains $K_e/8$ times as many responses as a full-size pool, this forward costs $75F/\rho$ units over the explorer pool. Our group-level score additionally requires one masked-image explorer forward per explorer response, costing $75F$ units in total. Other selection statistics used by the reproduced methods are by-products of sampling and add no sampling-side forward pass.

Under this normalization, the per-round sampling costs are

$$C_{\mathrm{OP}} = 100 + 100F, \tag{14}$$

$$C_{\mathrm{Eight}} = 100 + 100F, \tag{15}$$

$$C_{\mathrm{RR}} = 115 + 115F, \tag{16}$$

$$C_{\mathrm{Ours}} = 90 + (75/\rho + 90)\, F. \tag{17}$$

Here, OP denotes on-policy GRPO, Eight denotes the eight reproduced methods, and RR denotes random refresh. The first two regimes generate one full-size trainee pool per round, while random refresh additionally regenerates 15% of it. Our method instead builds the initial pool with the explorer (75 units) and regenerates only the selected 15% with the trainee.

Table 9: Per-round sampling cost, normalized to on-policy GRPO. Rollout generation counts generation only; modelled sampling time adds the trainee forward and the explorer's masked-image forward with $F = 0.1$.

| Cost | On-policy GRPO | Eight methods | Random refresh | Ours (1B→8B) | Ours (2B→8B) |
|---|---|---|---|---|---|
| Rollout generation units | 100 | 100 | 115 | 90 | 90 |
| Modelled sampling time | 100% | 100% | 115% | 144.5% | 117.3% |

**Per-Round Sampling Cost.** Table 9 instantiates the model. Counting generation only ($F = 0$), selective refresh costs 90 units per round, against 100 for on-policy GRPO and the eight methods and 115 for random refresh (Figure 3a). For modelled sampling time, we use $F = 0.1$ to reflect the substantially lower wall-clock cost of a batched teacher-forced forward relative to autoregressive decoding. Because the trainee forward covers the larger explorer pool, selective refresh then reaches 117.3% of on-policy GRPO for InternVL3-2B→8B (Figure 3b) and 144.5% for InternVL3-1B→8B, against 115% for random refresh; varying $F$ from $0.05$ to $0.2$ gives 104.3–140.0% and 118.6–190.0%, respectively.

The model intentionally accounts for sampling rather than the subsequent optimization pass. We analyze measured end-to-end wall-clock and total-training cost separately below.

### D.2 Measured and Cumulative Cost

**Measured Wall-Clock.** We complement the analytical sampling model with measured end-to-end wall-clock from completed training runs. The measured interval includes model loading, initial evaluation, eight training epochs, and final evaluation, since these components cannot be separated reliably from the available logs. A run enters the timing analysis only if it passes the validity criteria in Appendix C, completes all eight epochs, and records a final evaluation result. For each arm, we report the minimum wall-clock among valid completed runs to reduce contamination from queueing and transient GPU contention. All relative shares are computed against an on-policy run on the same evaluation frame whenever such a reference is available.

Table 10 summarizes the measured results. On ScienceQA, the pure selective-refresh arm takes 58% of the matched on-policy wall-clock with the 1B explorer and 40% with the 2B explorer. On AI2D, the corresponding shares are 63% and 35%. Across the remote HallusionBench and BLINK frames, the 2B-explorer configuration requires 52.7% and 40.8% of the matched on-policy wall-clock, respectively. The 1B-explorer configuration is more expensive because its larger $K = 48$ pool increases the trainee-side optimization cost, reaching 94.9% on HallusionBench and 96.2% on BLINK. Across all timed frames, including MMStar (29–39%) and the Qwen configurations (20–50%), these measurements give the 20–96% range reported in the main text.

**Timing Caveats.** The wall-clock measurements were collected on shared GPUs and therefore contain non-negligible scheduling and co-residency noise. Local runs use the interval from log creation to final write, while remote runs use the first in-pod GPU probe as the start time to exclude external queueing. The 1B-explorer runs also optimize over a larger $K = 48$ pool than either the 2B-explorer setting ($K = 24$) or on-policy GRPO ($K = 6$), which increases the training-side cost independently of rollout generation. This explains why their measured wall-clock shares are larger than those of the 2B-explorer runs. The reproduced eight methods use frozen pools and therefore provide a lower bound on their measured per-run time under our reproduction, rather than the cost of repeatedly sampling in their original training loops. The "Training only, 1B explorer pool" row is slower than "Ours, 1B explorer" although it never regenerates: without refresh, every group keeps all 48 explorer responses, so more groups retain reward variance and the run takes more optimizer steps (184 versus 134–138 in our logs), whereas refreshed groups shrink to six trainee responses and often lose reward variance, in which case GRPO skips them. We consequently interpret Table 10 as an empirical wall-clock range rather than a hardware-normalized systems benchmark.

**Cumulative Generation Cost.** We additionally report the cumulative rollout-generation cost over all eight training epochs. The eight reproduced methods generate their frozen pool once, giving a cumulative cost of 100 normalized units. Conventional random refresh starts from a trainee-sampled

Table 10: Measured end-to-end wall-clock for valid runs. Minutes and relative cost are reported against the matched on-policy reference for each evaluation frame.

| Frame | Arm | Minutes | Runs | Share |
|---|---|---|---|---|
| ScienceQA | On-policy GRPO | 334 | 1 | 100% |
| | Eight methods (median) | 79 | 8 | 24% |
| | Random refresh, 8B pool | 100 | 2 | 30% |
| | Ours, 1B explorer | 194 | 4 | 58% |
| | Ours, 2B explorer | 133 | 1 | 40% |
| | Random refresh, 1B explorer pool | 191 | 4 | 57% |
| | Training only, 1B explorer pool | 322 | 2 | 96% |
| AI2D | On-policy GRPO | 354 | 2 | 100% |
| | Eight methods (median) | 96 | 8 | 27% |
| | Random refresh, 8B pool | 90 | 2 | 25% |
| | Ours, 1B explorer | 222 | 5 | 63% |
| | Ours, 2B explorer | 125 | 2 | 35% |
| | Random refresh, 2B explorer pool | 129 | 2 | 36% |
| HallusionBench | On-policy GRPO | 233 | 3 | 100% |
| | Eight methods (median) | 44 | 24 | 18.9% |
| | Random refresh, 8B pool | 60 | 3 | 25.5% |
| | Ours, 1B explorer | 221 | 6 | 94.9% |
| | Ours, 2B explorer | 123 | 6 | 52.7% |
| BLINK | On-policy GRPO | 258 | 3 | 100% |
| | Eight methods (median) | 54 | 24 | 20.9% |
| | Random refresh, 8B pool | 92 | 3 | 35.8% |
| | Ours, 1B explorer | 248 | 6 | 96.2% |
| | Ours, 2B explorer | 105 | 6 | 40.8% |
| MMStar | Ours, 1B explorer | 176 | 3 | 39% |
| | Ours, 2B explorer | 131 | 2 | 29% |
| Qwen3.5-9B, DocVQA | Ours | 322 | 2 | 33% |
| Qwen3.5-9B, MathVista | Ours | 286 | 2 | 20% |
| Qwen3-VL-8B, DocVQA | Ours | 282 | 3 | 36% |
| Qwen3.5-9B, HallusionBench | Ours | 354 | 8 | 50% |
| Qwen3.5-9B, BLINK | Ours | 312 | 5 | 42% |

pool and refreshes 15 units in each of epochs 2–8, for a total of 205 units. Our configuration starts from the 75-unit explorer pool and adds the same 15-unit refresh in epochs 2–8, for a total of 180 units, of which 105 are trainee generation. On-policy GRPO regenerates the full pool at every epoch and therefore reaches 800 units. Relative to this on-policy total, the corresponding generation-only shares are 12.5%, 25.6%, 22.5% (13.1% from the trainee), and 100% for the eight reproduced methods, random refresh, our method, and on-policy GRPO, respectively.

The cumulative view uses the same nominal accounting as the per-round model. It should be interpreted separately from total training cost, since the optimization pass depends strongly on the per-configuration group size $K$.

### D.3 Scope of the Efficiency Claim

**As-Run Configurations.** The evaluated regimes do not use identical rollout-group sizes. Our explorer-sampled pools use $K = 6/\rho$, giving $K = 48$ for InternVL3-1B→8B and $K = 24$ for InternVL3-2B→8B, while the remaining explorer–trainee configurations use $K = 12$–$13$. The eight reproduced methods use a static trainee-sampled pool with $K = 8$, whereas on-policy GRPO uses $K = 6$ and regenerates the full pool at every epoch. The sampling-cost model above uses these as-run explorer pool sizes and takes the eight-response trainee pool as the full-size reference.

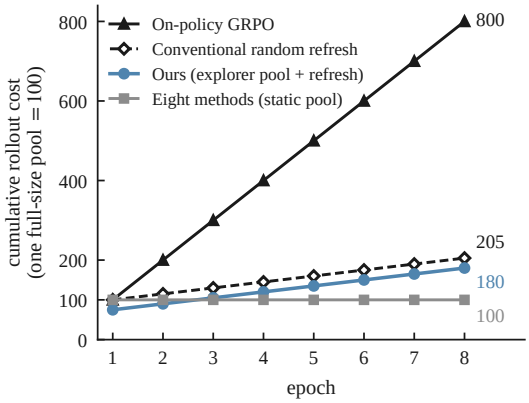


Figure 6: Cumulative rollout-generation cost over eight epochs. Costs are normalized so that one full-size trainee-sampled pool (8 responses per prompt) equals 100 units; teacher-forced forward passes are excluded.

The larger explorer-sampled pools deliberately trade additional trainee-side optimization for less full-size autoregressive generation.

**Training FLOPs.** Selective refresh does not necessarily reduce total training FLOPs. Because the trainee optimizes over $K = 12$–$48$ responses per rollout group, compared with $K = 6$ for on-policy GRPO, the optimization pass can offset or exceed the savings from reduced generation. We therefore make no claim of lower total FLOPs.

**What Selective Refresh Saves.** The primary efficiency benefit is a reduction in autoregressive rollout generation by the full-size trainee. This is the sequential, memory-bound component targeted by the small-explorer/partial-refresh design. Accordingly, our efficiency claims concern rollout generation cost and measured end-to-end wall-clock, rather than total training FLOPs; the modelled sampling time, which adds the forwards over the larger explorer pool, stays close to that of random refresh for 2B→8B (117.3% versus 115%) but reaches 144.5% for 1B→8B. Under this scope, selective refresh substantially reduces full-size rollout generation while remaining competitive with on-policy GRPO in the evaluated settings.

## E Training and Evaluation Details

**Training Setup.** All arms fine-tune the trainee with LoRA adapters of rank 16 and $\alpha = 32$ on the attention projections (for Qwen3.5, also on the input and output projections of its linear-attention layers), with a constant learning rate of $10^{-5}$, gradient accumulation over eight steps, and an entropy bonus of $0.01$. Each run trains on 200 prompts for eight epochs. Rollouts are sampled with temperature $0.9$ and top-$p$ $0.95$ (temperature $1.0$ and top-$k$ $40$ for Qwen3.5), with at most 896 generated tokens. The reward is binary answer correctness under the scoring rule below; groups whose responses all receive the same reward carry no gradient and are skipped. Masked-image statistics replace the input image with a uniform gray image (RGB value 127) of the input size ($448 \times 448$ for InternVL3). Evaluation uses greedy decoding with the same length limit.

**Answer Scoring.** The training reward and all reported accuracies use the same rule-based scorer. It extracts the final answer from the response (a boxed answer, an option letter, a number, or yes/no) and compares it with the gold answer after case and punctuation normalization; numeric answers match within $10^{-2}$, and for the free-text benchmarks (DocVQA, InfoVQA, and TextVQA) a word-boundary match between the gold answer and the response counts as correct. We do not use the official benchmark metrics, such as ANLS for DocVQA, and most evaluation slices are carved subsets rather than official test splits, so absolute accuracies are not comparable to published leaderboard numbers. All comparisons in the paper are between arms scored identically on the same evaluation slice.

**Terminology.** *Carved slice.* For a benchmark without a separate training split, the 200 training prompts are drawn from the benchmark and the evaluation slice consists of the remaining items, excluding any item that shares its index, its normalized question text, or its image (by MD5 hash) with a training prompt; training and evaluation are therefore disjoint at the index, question, and image level. *Evaluation frame.* An evaluation slice together with its initial accuracy $a_0$; arms are compared only within the same frame, and when the slice of a model–benchmark pair changed during the study, each version forms a separate frame. *Venue.* A benchmark evaluation slice, as listed in Table 6. *Cell.* One method–benchmark comparison against the random-refresh reference in Finding 1. *Unit.* One explorer–trainee configuration, benchmark, and frame in the selective-refresh evaluation (Table 7). *Strongest valid seed.* The run with the highest final accuracy among the runs of an arm that pass the validity criteria in Appendix C.