WHEN TO STOP REUSING: DYNAMIC GRADIENT GATING FOR SAMPLE-EFFICIENT RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has become the dominant paradigm for advanced reasoning in Large Language Models (LLMs), but rollout samples are expensive to obtain, making sample efficiency a critical bottleneck. A natural remedy is to reuse each rollout batch for multiple gradient updates, a standard practice in classical RL. Yet in RLVR, this amplifies policy shift, leading to severe performance degradation. Detecting the onset of degradation early enough to stop reuse remains an open and challenging problem. We close this gap by identifying the *Disproportionate Weight Divergence (DWD)* phenomenon: performance degradation is synchronized with a sharp surge in the `lm_head` weight change, while intermediate layers remain stable. Empirically, we verify that DWD emerges consistently across diverse LLMs and tasks. Theoretically, we establish that (i) harmful gradients enter the `lm_head` gradient directly, while their contributions to intermediate layers are substantially attenuated through downstream Jacobian projections; and (ii) the `lm_head` gradient norm lower-bounds policy divergence. These results establish the `lm_head` gradient norm as a principled, real-time signal of catastrophic policy shift. Guided by this insight, we propose *Dynamic Gradient Gating (DGG)*, a lightweight intervention that monitors the `lm_head` gradient norm in real time and intercepts harmful gradients before they corrupt the optimizer. DGG consistently matches or exceeds the standard single-use baseline, achieving up to $2.93\times$ sample efficiency and $2.14\times$ wall-clock speedup across math, ALFWorld, WebShop, and search-augmented QA tasks.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has become the key driver for unlocking advanced reasoning in Large Language Models (LLMs), powering recent breakthroughs in long-horizon mathematics, code, and agentic tasks (Shao et al., 2024; Guo et al., 2025; Yu et al., 2025). However, this powerful paradigm introduces a severe computational bottleneck (Wu et al., 2025; Team et al., 2025). In modern RLVR pipelines, rollout generation dominates the training budget, often consuming over 80% of post-training GPU hours (Arnal et al., 2026). As models scale and tasks demand longer horizons, improving the sample efficiency of rollout utilization has emerged as a central prerequisite for sustainable training.

A natural approach to improving sample efficiency is *sample reuse*, which performs multiple gradient updates on each fresh rollout batch. This technique is well-established in policy-gradient RL (Schulman et al., 2017; 2015) and has driven substantial efficiency gains in classical domains. Unfortunately, as extensively observed in recent RLVR systems, naive sample reuse exacerbates the policy shift between the updating policy and the behavior policy, triggering catastrophic *training collapse* (Zheng et al., 2025; Yang et al., 2025b; Zheng et al., 2026a; Liang et al., 2026; Baroian & Berger, 2026)—manifesting as severe *performance degradation*. Consequently, production pipelines have overwhelmingly reverted to a conservative *single-use rollout regime*, where each rollout is strictly discarded after one update, severely underutilizing each sample despite its steep rollout cost. Motivated by this challenge, we aim to investigate the following research question:

> *How can we reliably detect the underlying signals of policy shift in real time—before it triggers training collapse—to unlock the full efficiency gains of sample reuse?*

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.

To answer this question, we begin with an empirical investigation into the microscopic dynamics of sample-reuse-induced collapse. Our profiling reveals a structural asymmetry that we term the *Disproportionate Weight Divergence (DWD)* phenomenon: **performance degradation is synchronized with a sharp surge in the `lm_head` weight change, while intermediate layers remain stable**. This observation highlights the `lm_head` as a natural target for collapse monitoring and motivates a theoretical analysis of its distinctive gradient behavior.

We characterize two structural properties underlying this phenomenon. First, harmful gradients **enter the `lm_head` directly**, while their intermediate-layer contributions are **substantially attenuated** through downstream Jacobian projections (Theorem 1); the architectural factor in the resulting bound is empirically below 10^{-1} at the median across diverse LLMs (Table 2). Second, we establish a lower-bound relation between the `lm_head` gradient norm and the empirical Pearson χ^2 divergence between the updating and behavior policies (Theorem 2). Together, these results provide a mechanistic account of DWD and motivate the `lm_head` gradient norm as a monitoring signal.

Together, these empirical and theoretical findings motivate the `lm_head` gradient norm as a principled, real-time monitoring signal for catastrophic policy shift. Guided by this insight, we propose *Dynamic Gradient Gating (DGG)*, a lightweight intervention that detects anomalous gradient surges, **discards harmful gradients**, and **terminates reuse** before they corrupt the model and optimizer.

We demonstrate the widespread presence of the DWD phenomenon across **eleven LLMs** from Qwen (Yang et al., 2025a) and Llama (Touvron et al., 2023) families (1.5B–14B parameters) and **four diverse tasks**, covering mathematical reasoning and agentic tasks (WebShop, ALFWorld, and search-augmented QA). Across **ten benchmarks**, DGG matches or exceeds the strict single-use baseline, achieving up to **2.93**$\times$ sample efficiency and **2.14**$\times$ wall-clock speedup. These results confirm that the efficiency potential of sample reuse can be fully unlocked with a principled, lightweight intervention. Our main contributions are as follows:

- **Empirical Discovery.** We identify the DWD phenomenon, where sample reuse-induced performance degradation is structurally synchronized with a localized weight divergence at the `lm_head`.
- **Theoretical Analysis.** We theoretically account for DWD by establishing a structural gradient asymmetry and showing that the `lm_head` gradient norm lower-bounds policy divergence.
- **Algorithm.** We propose DGG, a lightweight mechanism that monitors the `lm_head` gradient norm via a Z-score test and intercepts harmful gradients before they corrupt the optimizer.
- **Experiments.** We validate the broad presence of DWD across diverse LLMs and tasks, and demonstrate that DGG achieves up to 2.93$\times$ sample efficiency without sacrificing final performance.

2 Related Work

RLVR. RLVR is the dominant paradigm for unlocking LLM reasoning (Guo et al., 2025; Yu et al., 2025; Shao et al., 2024), yet its rollout stage severely dominates training time (Zheng et al., 2026b;a; Zhang et al., 2026a). Concurrently, scaling rollout exploration remains highly effective for improving performance across diverse tasks (Li et al., 2025a; Xing et al., 2026; Dong et al., 2026; Cao et al., 2025). Given that rollouts are both *expensive and highly valuable*, failing to fully exploit the latent learning value of each rollout has become a critical bottleneck to scaling RLVR.

Sample-Efficient RLVR. Existing efforts improve sample efficiency at the *rollout stage* through experience replay (Arnal et al., 2026; Sun et al., 2026; Wang et al., 2025a; Zhan et al., 2026) or adaptive allocation based on prompt difficulty (Liao et al., 2025; Wang et al., 2025a; Li et al., 2025b) and variance (Zheng et al., 2026b; Zhang et al., 2026b; Nguyen et al., 2026). However, both still operate within the *single-use rollout regime*. At the *update stage*, SFPO (Wang et al., 2026) rescales the accumulated update via interpolation after a fixed number of reuse steps, but *neither detects nor filters harmful gradients as they emerge*, leaving each batch to follow a globally predefined reuse count even when additional updates become harmful; see Appendix H for detailed comparison.

Our Focus. We focus on a different control question at the *update stage*: given each rollout batch, *how many gradient updates can be safely performed before further reuse becomes harmful?* The key is to detect harmful updates in real time and terminate current-batch reuse before they are applied, a per-batch control problem that remains underexplored in existing methods.

3 PRELIMINARIES: GRPO AND SAMPLE REUSE

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) optimizes the policy π_θ by maximizing a clipped surrogate objective over T active tokens:

$$\mathcal{L}_{GRPO}(\theta) = \frac{1}{T}\sum_{i=1}^{T} \min\Big(r_i(\theta)\hat{A}_i,\ \text{clip}\big(r_i(\theta),\, 1-\epsilon_c,\, 1+\epsilon_c\big)\hat{A}_i\Big), \tag{1}$$

where $\hat{A}_i$ is the group-normalized advantage and $r_i(\theta) \triangleq \frac{\pi_\theta(a_i|\cdot)}{\pi_{old}(a_i|\cdot)}$ is the importance ratio against the behavior policy π_{old} for the sampled token a_i. Crucially, for a token i whose surrogate term lies on the unclipped branch, its gradient contribution is

$$g_i = r_i(\theta)\hat{A}_i \nabla_\theta \log \pi_\theta(a_i \mid \cdot). \tag{2}$$

This formulation highlights that the token-level gradient signal is multiplicatively scaled by $r_i(\theta)$.

The Vulnerability of Sample Reuse in LLMs. The gradient formulation in Eq. 2 reveals why sample reuse, a cornerstone of classical RL (Schulman et al., 2017), breaks down for LLMs. The vast action space and long reasoning horizons force the behavior policy π_{old} to assign *vanishingly small probabilities* to numerous tail tokens. Since $r_i(\theta)$ uses these tiny probabilities as denominators, successive reuse steps create a feedback loop in which modest policy shifts compound rapidly, inflating $r_i(\theta)$ by orders of magnitude (Chen et al., 2025; Zheng et al., 2025; Wang & Gao, 2025). Because $r_i(\theta)$ enters Eq. 2 as a multiplicative scalar, this inflation disproportionately magnifies the gradients of these tokens, **letting sparse tail tokens dominate the update and drive the model to collapse** (Yang et al., 2026; Liu et al., 2026). The standard clipping mechanism in Eq. 1 does not prevent this. Its protection is asymmetric: tokens with negative advantages and highly inflated ratios ($r_i(\theta) > 1+\epsilon_c$) remain unclipped (Ye et al., 2020; Wang et al., 2025b; Yu et al., 2025), so anomalously large gradients propagate freely into the model.

While this vulnerability is widely observed (Zheng et al., 2026a; Liang et al., 2026), its underlying mechanism remains unclear. We investigate it empirically and theoretically in the next section.

4 DISPROPORTIONATE WEIGHT DIVERGENCE (DWD)

Having established in Section 3 why sample reuse destabilizes RLVR training, we now seek an indicator signal inside the network that reliably reflects this instability in real time. Section 4.1 identifies such a signal empirically through microscopic profiling—the Disproportionate Weight Divergence (DWD) phenomenon—and Section 4.2 theoretically explains why DWD arises.

4.1 EMPIRICAL OBSERVATION OF THE DWD PHENOMENON

Profiling Setup. To characterize sample-reuse-induced collapse, we profile eleven LLMs from the Llama and Qwen families (1.5B–14B) across four tasks: mathematical reasoning, WebShop, ALFWorld, and search-augmented QA. We compare GRPO w/o Reuse against GRPO w/ Naive Reuse (fixed updates per rollout batch), tracking (i) task performance and (ii) relative weight changes grouped by component type (`lm_head`, `attn`, `mlp`). Throughout the paper, one *RL step denotes one fresh rollout batch*; sample-reuse methods may perform multiple updates per step, while methods at *the same RL step share the same rollout budget*.

Core Finding. Figure 1 shows representative results. Without sample reuse, performance improves steadily and weight changes remain smooth across all layers. With sample reuse, convergence is substantially faster at first but soon gives way to a severe performance degradation, accompanied by a structurally asymmetric pattern in the layer-wise weight dynamics, which we formalize as follows.

> **Observation 1** (Disproportionate Weight Divergence)**.** *During sample-reuse RLVR training, performance degradation is synchronized with a sharp surge in the* `lm_head`*'s weight change, while intermediate layers (attention and MLP) remain stable.*

This pattern holds across all profiled settings (see Appendix D). Appendix J further shows that without sample reuse, the `lm_head` surge does not emerge even over the full training trajectory, confirming that sample reuse itself is the direct cause, rather than the increased number of updates.

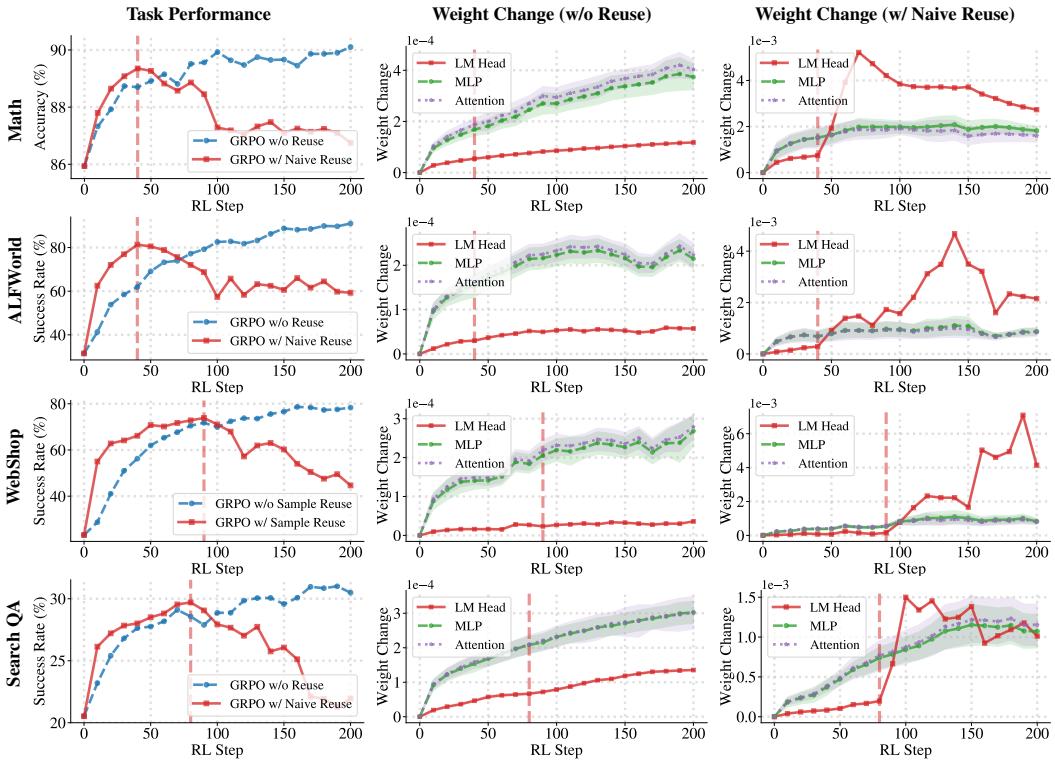

Figure 1: **Illustration of the Disproportionate Weight Divergence (DWD) phenomenon** on Qwen3-4B-Instruct across various tasks. Relative weight change is defined as $\|W_t - W_{t-\Delta}\|_F / \|W_{\text{ref}}\|_F$ (W_{ref}: initial pretrained weight; Δ: profiling interval, set to 10 RL steps); red dashed lines mark the onset of performance degradation for GRPO w/ Naive Reuse. Observation: *Sample reuse accelerates early convergence, followed by a severe performance degradation synchronized with an abrupt, isolated surge in the LM Head, while intermediate layers (Attention and MLP) remain stable.* Consistent results across the other six LLMs are provided in Appendix D.

4.2 Theoretical Analysis of the DWD Phenomenon

The empirical findings in Section 4.1 raise two questions: *why is the gradient anomaly structurally localized to the* `lm_head`*, and why is the* `lm_head` *gradient norm theoretically linked to the policy shift that drives training collapse??* We answer these two questions via a *Structural Gradient Asymmetry* (Theorem 1, Section 4.2.1) and a *Divergence Lower Bound* (Theorem 2, Section 4.2.2), respectively.

Setup and notation. For any active token position i, let $h_{L,i} \in \mathbb{R}^{d_{\text{model}}}$ denote the post-norm hidden state entering the `lm_head`, and $x_i^{\text{int}} \in \mathbb{R}^{d_{\text{model}}}$ the input to an arbitrary intermediate linear layer. Let a_i be the sampled token, $\pi_\theta(\cdot \mid h_{L,i})$ and π_{old} the current and behavior policies, $r_i \triangleq \pi_\theta(a_i)/\pi_{\text{old}}(a_i)$ the importance sampling ratio, and $\hat{A}_i$ the group-normalized advantage. Let J_i denote the composite Jacobian mapping the intermediate linear layer's output at position i to the final logits. We measure gradient magnitude using the Frobenius norm $\|M\|_F$, and policy divergence using the Pearson χ^2.

4.2.1 Structural Gradient Asymmetry

Our analysis proceeds in three steps: (1) decomposing the per-layer gradients to isolate the error signal, (2) establishing regularity conditions on the forward and backward signals, and (3) deriving a quantitative bound on the inter-layer gradient attenuation. We then verify this bound empirically across multiple LLM architectures. The following proposition formalizes the gradient decomposition.

Proposition 1 (Gradient Decomposition)**.** *Let $\mathcal{L}_i$ denote the objective for token a_i, and define the error signal as $E_i \triangleq r_i \hat{A}_i (e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})) \in \mathbb{R}^{d_{vocab}}$. The* `lm_head` *gradient and the gradient contribution through any intermediate linear layer at position i admit the following rank-1 forms:*

$$G_i^{lm} \triangleq E_i \, h_{L,i}^\top, \qquad G_i^{int} \triangleq \left(J_i^\top E_i\right)\left(x_i^{int}\right)^\top, \tag{3}$$

where $J_i = \partial z_i / \partial y_i$ is the Jacobian of logits z_i with respect to the intermediate output $y_i = W_{int} x_i^{int}$.

Proposition 1 (proof in Appendix A.1) shows that the `lm_head` absorbs the error signal E_i directly, whereas G_i^{int} is modulated by the Jacobian projection $J_i^\top$. Since $\|uv^\top\|_F^2 = \|u\|_2^2 \cdot \|v\|_2^2$ for any rank-1 matrix, this structure cleanly separates the *error signal* (E_i or $J_i^\top E_i$) from the *representation signal* ($h_{L,i}$ or x_i^{int}), allowing each factor to be bounded independently. We first bound the magnitude of the representation signals feeding the respective layers.

Lemma 1 (Bounded Activations under Pre-Norm Architectures). *For any active token i, let $h_{L,i}$ denote the* `lm_head` *input and x_i^{int} denote the intermediate-layer input. There exist strictly positive constants $\alpha_{\min}$ and $\beta_{\max}$ such that:*

$$\|h_{L,i}\|_2^2 \;\geq\; \alpha_{\min} \cdot d_{model}, \qquad \|x_i^{int}\|_2^2 \;\leq\; \beta_{\max} \cdot d_{model}. \tag{4}$$

The upper bound $\beta_{\max}$ holds uniformly across all intermediate linear layers in standard Pre-Norm Transformers, covering both layers fed directly by RMSNorm (e.g., QKV and FFN up-projections) and those fed by internal sub-network outputs (e.g., attention output and FFN down-projections). See Appendix A.2 for the layer-by-layer derivation. We next bound the Jacobian-projected error signal in the backward pass.

Assumption 1 (Bounded Logit Sensitivity). *There exists a strictly positive constant C, such that for any token position i:*

(i) $\mathbb{E}_{j\sim\pi_\theta}\|(J_i)_{j,:}\|_2^2 \leq C$, *bounding the expected logit sensitivity to intermediate perturbations;*

(ii) $\|(J_i)_{a_i,:}\|_2^2 \leq C$, *extending the same bound to the sampled token a_i.*

This bounded-Jacobian property is well-studied in deep learning theory, both as a sufficient condition for generalization (Sokolić et al., 2017) and as an explicit regularization target (Yoshida & Miyato, 2017). We further empirically verify that this assumption holds across diverse LLMs, and that $\mathcal{C}_{\text{struct}}$ defined in Theorem 1 as a whole remains stable throughout RL training, including at the onset of training collapse (Appendix A.4).

Combining the gradient decomposition (Proposition 1), forward activation bounds (Lemma 1), and Jacobian regularity (Assumption 1), we formally establish the gradient asymmetry between the `lm_head` and intermediate layers, culminating in our first main result.

Theorem 1 (Structural Gradient Asymmetry). *Under Lemma 1 and Assumption 1, for any sampled token a_i with policy probability $\pi_\theta(a_i) < 1$ and non-zero advantage, we have:*

$$\frac{\|G_i^{int}\|_F^2}{\|G_i^{lm}\|_F^2} \;\leq\; \mathcal{C}_{struct} \cdot \frac{1}{(1-\pi_\theta(a_i))^2}, \tag{5}$$

where $\mathcal{C}_{struct} \triangleq \frac{4\beta_{\max} C}{\alpha_{\min}}$ is a strictly positive architectural constant.

Theorem 1 factorizes this asymmetry into two multiplicative terms: the architectural constant $\mathcal{C}_{\text{struct}}$, which captures all architecture-dependent quantities and is empirically small across diverse LLMs (Section A.4); and the policy-confidence factor $(1-\pi_\theta(a_i))^{-2}$, governed by how concentrated π_θ is on a_i. The proof is deferred to Appendix A.3.

Quantitative instantiation of the bound. We measure all constants entering $\mathcal{C}_{\text{struct}}$ directly from forward-pass activations and Jacobian estimates across multiple LLMs spanning the Llama, Qwen, and GPT-OSS families (1.5B–235B parameters; full table in Appendix A.4). The factor is uniformly small across all tested architectures—below 10^{-1} at the median and below 0.5 even at the 95th-percentile worst case. Moreover, $\mathcal{C}_{\text{struct}}$ remains stable throughout RL training, including at the onset of sample-reuse-induced collapse (Appendix A.4). These measurements quantitatively ground the structural attenuation characterized by Theorem 1.

Connection to the DWD phenomenon. Sample reuse in LLMs risks training collapse when the gradient is dominated by tail tokens, i.e., those with vanishingly small probability under π_{old} but massively inflated importance ratios r_i and large advantages $|\hat{A}_i|$ (Section 3). Theorem 1 shows that the bound on intermediate-layer gradients is tightest on exactly these tokens: their low policy confidence drives $(1-\pi_\theta(a_i))^{-2}$ toward its minimum, and the architectural constant $\mathcal{C}_{\text{struct}}$ is empirically small ($< 10^{-1}$ at the median). *When tail tokens dominate, this structural gradient asymmetry strongly favors the* `lm_head`*, providing a mechanistic account of the DWD phenomenon.*

4.2.2 GRADIENT-BASED POLICY-DIVERGENCE LOWER BOUND

Our analysis proceeds in two steps: (1) relating the `lm_head` gradient norm to the empirical second moment of importance ratios, and (2) connecting this moment to an empirical Pearson χ^2 statistic.

Aggregating the per-token gradient G_i^{lm} over T active tokens, we define the batch-level `lm_head` gradient $G^{\text{lm}} \triangleq \frac{1}{T}\sum_{i=1}^{T} G_i^{\text{lm}}$, matching the gradient propagated through the GRPO objective (Eq. 2).

Lemma 2 (Gradient Norm via Importance-Ratio Second Moment). *Let $\overline{r^2} \triangleq \frac{1}{T}\sum_{i=1}^{T} r_i^2$ denote the empirical second moment of importance ratios. Then*

$$\|G^{lm}\|_F^2 \;\le\; c_{\max}\,\overline{r^2}, \qquad \textit{where}\;\; c_{\max} \triangleq \max_i \hat{A}_i^2\,\|e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\|_2^2\,\|h_{L,i}\|_2^2. \tag{6}$$

We now relate $\overline{r^2}$ to the Pearson χ^2 statistic, yielding our second main result.

Theorem 2 (`lm_head` Gradient Lower-Bounds Policy Divergence). *Let $\hat{\chi}^2 \triangleq \frac{1}{T}\sum_{i=1}^{T}(r_i^2 - 1)$ denote the empirical estimator of the Pearson χ^2 divergence $\chi^2(\pi_\theta \| \pi_{old})$. Then*

$$\|G^{lm}\|_F^2 \;\le\; c_{\max}\big(1 + \hat{\chi}^2\big), \quad \textit{or equivalently,} \quad \hat{\chi}^2 \ge \|G^{lm}\|_F^2 / c_{\max} - 1. \tag{7}$$

The proofs of Lemma 2 and Theorem 2 are deferred to Appendix B.

Notably, $c_{\max}$ captures the advantage, output-distribution, and representation scales without any explicit importance-ratio factor, whereas $1 + \hat{\chi}^2 = \frac{1}{T}\sum_i r_i^2$ directly captures the empirical second moment of the importance ratios and can grow rapidly under sample reuse. Theorem 2 thus establishes a structural link between the `lm_head` gradient energy and off-policy policy shift, motivating the `lm_head` gradient norm as the monitoring signal. Importantly, this bound provides a *structural justification for signal selection*, rather than a calibrated numerical certificate of collapse.

5 DYNAMIC GRADIENT GATING (DGG)

Together, the DWD phenomenon (Observation 1) and our theoretical analysis motivate the `lm_head` gradient norm as a principled monitoring signal of catastrophic policy shift: Theorem 1 provides a mechanistic account of the structural asymmetry underlying DWD, while Theorem 2 connects the `lm_head` gradient norm to policy divergence. This empirical-theoretical convergence motivates a simple yet principled intervention—*Dynamic Gradient Gating* (DGG)—that monitors this localized signal in real time and intercepts harmful updates *before* they corrupt the weights and optimizer state.

Online detection via Z-score test. Since the magnitude of $\|G^{\text{lm}}\|_F^2$ varies across models, tasks, and training stages, we adopt the classical Z-score test, which adaptively standardizes against its recent statistics. Let $g_t \triangleq \|G_t^{\text{lm}}\|_F^2$ denote the `lm_head` gradient energy at step t, and $\Delta g_t \triangleq g_t - g_{t-1}$ its step-wise increment. Since DWD manifests as an abrupt surge rather than sustained high magnitude, we monitor Δg_t, isolating fast changes from slow drift. Maintaining running estimates of the mean μ_t and standard deviation σ_t of Δg_t, we compute the instantaneous Z-score:

$$z_t \;\triangleq\; \frac{\Delta g_t - \mu_t}{\sigma_t + \varepsilon}. \tag{8}$$

A step is flagged as anomalous whenever $z_t > \tau$, where τ is a tunable threshold.

Pre-optimizer gradient interception. Once $z_t > \tau$, DGG triggers a two-stage intervention: *(i) Gradient Discard.* We zero out the gradient *before* Adam, preventing corruption of its moment estimates that would persist for many steps. *(ii) Reuse Termination.* We terminate the reuse loop and draw a fresh batch, breaking the feedback loop where a drifted π_θ keeps inflating importance ratios. This pre-optimizer design stops the harm at its source (pseudocode in Appendix C; implementation details, including running statistics formulation and hyperparameter settings, in Appendix I.2).

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate DGG across mathematical reasoning and agentic tasks. For reasoning, we use MATH500 (Lightman et al., 2024), AIME25 (Zhang & Math-AI, 2025), Minerva

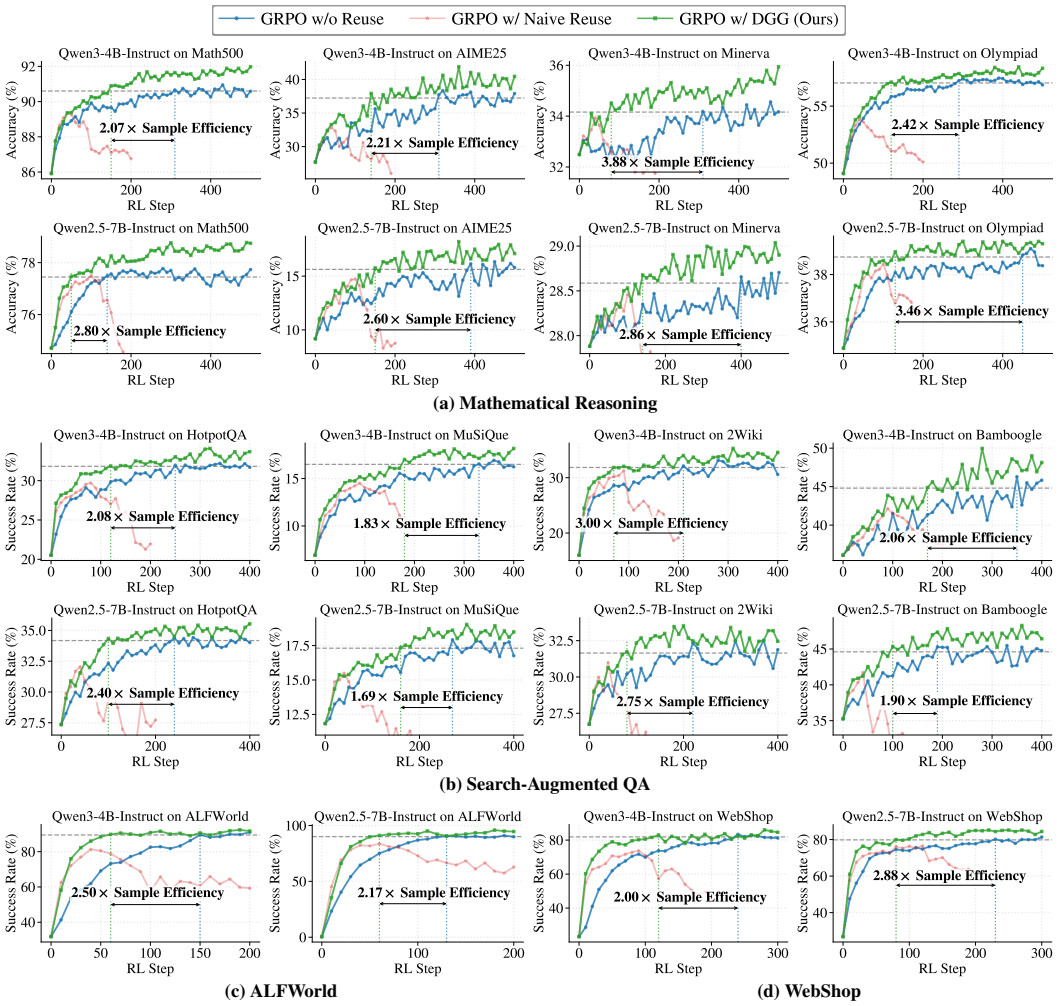

Figure 2: **Performance comparison under matched rollout budgets** across two LLMs and four tasks. Each RL step corresponds to one fresh rollout batch. GRPO w/ Naive Reuse uses a fixed number of updates per batch, equal to DGG's maximum reuse. Gray dashed lines mark the converged performance of GRPO (Single-Use Rollout), averaged over the last five checkpoints, as the reference for sample efficiency. Observation: *DGG avoids the instability observed under naive sample reuse, substantially improving sample efficiency without sacrificing final performance.* Appendix E extends DGG to Llama models, DAPO, and models up to 14B, providing further evidence of generalization.

Math (Lewkowycz et al., 2022), and Olympiad Bench (He et al., 2024), reporting mean@16 accuracy. For agentic evaluation, we consider ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and search-augmented QA on HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023).

Training details. We use Qwen3-4B-Instruct-2507 and Qwen2.5-7B-Instruct for the main GRPO experiments, with additional evaluations on Llama-3.2-3B-Instruct under GRPO and Qwen2.5-7B/14B-Instruct under DAPO. We set the maximum reuse $K = 4$, and select τ from $\{0.1, 0.5, 1.0\}$. *Each RL step denotes one fresh rollout batch*, so *methods at the same step share the same rollout budget* despite different numbers of policy updates per batch. Full configurations are in Appendix I.

6.2 Main Results

We compare DGG against vanilla GRPO with single-use rollouts (the standard practice in current RLVR systems) and GRPO with naive sample reuse (the most direct alternative). Figure 2 reports the

[2]Speedup factor is defined as baseline cost/DGG cost, and relative reduction as $1 -$ DGG cost/baseline cost, where cost is the total rollouts (or wall-clock time) needed to reach the baseline's converged performance.

Table 1: **Rollout speedups (sample efficiency) and wall-clock speedups of DGG** across two models and four tasks. Speedups are relative to the single-use GRPO baseline's converged performance (last five checkpoints averaged); parentheses show relative reductions[2]. Math and Search-QA average four datasets each. Observation: *DGG achieves up to* $2.93\times$ *rollout and* $2.14\times$ *wall-clock speedups.* See Appendix G.2 for a detailed runtime breakdown, showing that rollout-generation savings outweigh the additional policy-update cost.

Model	Math (avg.)		ALFWorld		WebShop		Search-QA (avg.)	
	Rollout	Time	Rollout	Time	Rollout	Time	Rollout	Time
Qwen3-4B-Instruct	2.64× (62.2%)	1.45× (30.9%)	2.50× (60.0%)	2.14× (53.4%)	2.00× (50.0%)	1.31× (23.6%)	2.24× (55.4%)	1.82× (45.1%)
Qwen2.5-7B-Instruct	2.93× (65.9%)	1.37× (26.9%)	2.17× (53.9%)	1.90× (47.5%)	2.88× (65.2%)	1.81× (44.7%)	2.19× (54.3%)	1.64× (38.9%)

training dynamics across all settings, while Table 1 quantifies the resulting savings in rollouts and wall-clock time; seed stability is further verified in Appendix F. We summarize main findings below.

DGG substantially mitigates the training instability induced by naive sample reuse. As shown in Figure 2, although GRPO w/ Naive Reuse converges faster than the single-use baseline in the early stage, it suffers from a consistent failure pattern: performance is severely degraded following the initial improvement, often before reaching the single-use baseline's final performance. In contrast, GRPO w/ DGG retains the early-stage acceleration while avoiding such collapses across all settings. This is consistent with our analysis in Section 4: by monitoring the `lm_head` gradient norm, whose connection to policy shift is characterized by Theorem 2, DGG discards harmful updates before they affect the optimizer state, addressing the main failure mode of sample reuse.

DGG significantly improves both sample and wall-clock efficiency without sacrificing final performance. Figure 2 shows that GRPO w/ DGG consistently reaches the single-use baseline's converged performance with substantially fewer RL steps across all 16 settings. Table 1 further quantifies these gains in terms of total training cost: DGG achieves $2.00\times$–$2.93\times$ rollout speedups and $1.31\times$–$2.14\times$ wall-clock speedups over the single-use baseline. Moreover, DGG shows modestly higher final performance in several settings, which is attributable to the additional optimizer updates enabled by safe sample reuse under the same rollout budget. The absolute runtime breakdown further shows that the reduction in rollout generation consistently outweighs the additional policy-update cost introduced by reuse (Appendix G.2).

The benefits of DGG generalize across models, tasks, and policy-optimization algorithms. The main results demonstrate consistent improvements across two Qwen models and four representative task domains. Appendix E further ***extends DGG to the Llama family, DAPO, and models up to 14B***, providing additional evidence of its generalization across model families, task domains, policy-optimization algorithms, and scales. This generality aligns with DWD's prevalence across architectures and tasks (Section 4), providing a principled foundation for DGG's broad applicability.

6.3 Analysis and Discussion

Beyond demonstrating DGG's effectiveness, we now examine three aspects of its design: the choice of `lm_head` gradient norm as the monitoring signal, the advantage of adaptive gating over fixed-reuse alternatives, and practical guidelines for setting its hyperparameters.

Comparison with alternative monitoring signals. Figure 3 evaluates four candidate monitoring signals: the KL divergence between π_θ and π_{old}, the clip ratio, the global gradient norm, and the `lm_head` gradient norm. While the KL divergence, clip ratio, and global gradient norm fail to exhibit a distinct spike at the onset of collapse, the `lm_head` gradient norm displays a sharp, well-scaled spike precisely synchronized with collapse onset. *This observation is consistent with the structural asymmetry characterized by Theorem 1 and the divergence lower bound established by Theorem 2, thereby justifying its selection as DGG's monitoring signal.*

Comparison between fixed and adaptive reuse mechanisms. A natural baseline for stability is to manually tune a conservative fixed-reuse mechanism. Figure 4 compares DGG against fixed-reuse variants with $K \in \{1, 2, 3, 4\}$. Fixed reuse exposes a fundamental trade-off: small counts maintain stability but limit efficiency, whereas larger counts accelerate early convergence but trigger collapse. By contrast, DGG dynamically modulates reuse based on the `lm_head` gradient, inheriting aggressive

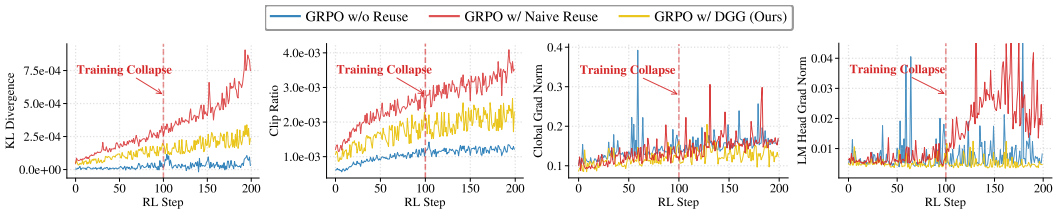

Figure 3: **Comparison of different monitoring signals** on Qwen2.5-7B-Instruct (Math500). The red dashed line marks the onset of training collapse for GRPO w/ Naive Reuse. Observation: *While KL divergence, clip ratio, and global gradient norm lack clear collapse signals, the* `lm_head` *gradient norm provides a sharp spike at collapse, serving as a reliable indicator that is consistent with our Structural Gradient Asymmetry (Theorem 1) and Divergence Lower Bound (Theorem 2).*

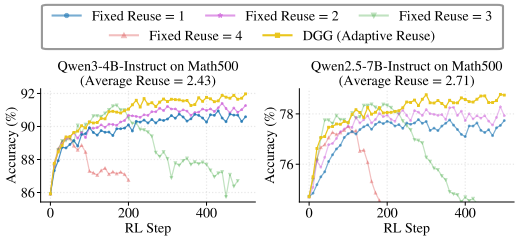

Figure 4: **Comparison between fixed and adaptive reuse mechanisms.** Observation: *The adaptive mechanism consistently performs better than all fixed counterparts across multiple reuse steps.*

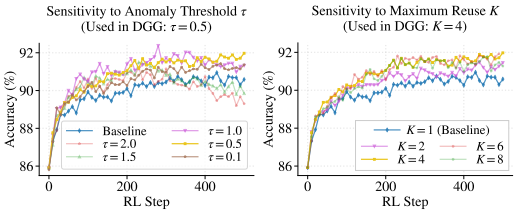

Figure 5: **Sensitivity analysis on DGG's hyperparameters τ and K**, conducted on Qwen3-4B-Instruct (Math500). Note that $K = 1$ reduces to the single-use baseline. Observation: *DGG achieves better performance than the single-use baseline across most settings.*

reuse's early acceleration while preserving stability via real-time gating. *DGG thus consistently outperforms all fixed counterparts, resolving the stability-efficiency trade-off intrinsic to fixed reuse.* The empirical reuse-count statistics further confirm this adaptive behavior: DGG selects different reuse depths across batches, tasks, models, and training stages (Appendix G.1).

Sensitivity analysis and practical recommendations. DGG introduces two hyperparameters: the anomaly threshold τ and the maximum reuse K, where $K = 1$ reduces to the single-use baseline. As shown in Figure 5, $\tau \in \{0.1, 0.5, 1.0\}$ achieves better performance compared with the single-use baseline, and *the cost of mis-tuning τ is asymmetric*: an overly small τ merely sacrifices a few reuse steps and still outperforms the single-use baseline, whereas an overly large τ allows harmful gradients to pass and degrades performance. We therefore recommend selecting τ from the candidate set $\{0.1, 0.5, 1.0\}$, biased toward the smaller end. For K, all settings with $K \geq 2$ outperform the single-use baseline, with marginal differences across $K \in \{4, 6, 8\}$. Since larger K incurs higher worst-case computation, we recommend $K = 4$ as a practical default.

7 Conclusion and Limitations

We identify DWD: during sample-reuse collapse, harmful gradients concentrate at the `lm_head`, whose abrupt divergence tracks performance degradation. We attribute this localization to downstream Jacobian attenuation and establish the `lm_head` gradient norm as a principled signal of catastrophic policy shift. DGG monitors this signal online and intercepts anomalous updates before they corrupt the optimizer. DWD generalizes across eleven LLMs and diverse tasks, while DGG achieves up to $2.93\times$ rollout efficiency and $2.14\times$ wall-clock speedup across ten reasoning and agentic benchmarks without performance loss, enabling safe, signal-driven reuse.

Limitation. Our experiments cover GRPO and DAPO, while our theoretical analysis extends to the clipped policy-gradient component of PPO. It does not cover gradients introduced by value-function optimization, including those propagated through a shared actor–critic backbone. A full empirical evaluation of actor–critic PPO remains future work. Due to the substantial computational cost of RL training, our end-to-end evaluations are limited to models up to 14B parameters; validating DWD and DGG at larger scales remains another important direction.

AI USE STATEMENT

In this work, we used generative AI tools for [assisting in the writing of proofs, proposing or refine hypotheses, implementing methods, and assisting with translation]. We have not used generative AI tools for [helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, designing or providing feedback on research methodology or experiments, cleaning and reformating dataset, supporting qualitative and thematic data analysis, and interpreting results], and [generating synthetic data sets] is not applicable to this work. Additionally, we used generative AI tools for [creating or modify scientific figures or images, summarizing or analyzing existing literature, identifying research topics or research gaps, editing research papers to improve readability, providing structural suggestions for research papers, and providing titles or keywords for research papers]. We have reviewed all AI-assisted work. [LLM-generated code was independently verified and tested for correctness by two authors; LLM-generated research ideas were checked for potential plagiarism through a manual literature survey; and LLM-generated proofs were carefully reviewed line by line by two authors.]. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work introduces Dynamic Gradient Gating (DGG), a lightweight training-stability mechanism for improving the sample and computational efficiency of RLVR. By reducing rollout and wall-clock costs, DGG may lower the energy consumption and computational barriers associated with LLM post-training, potentially making RL research more accessible beyond well-resourced organizations. As a general-purpose training method, DGG does not introduce new models, datasets, or application-specific capabilities. Nevertheless, models trained with DGG inherit the broader safety, fairness, and misuse risks associated with their underlying models, reward designs, and deployment settings. We do not identify additional application-specific risks introduced by DGG beyond those generally associated with LLM training, and encourage practitioners to pair efficiency improvements with appropriate safeguards for responsible model development and deployment.

REPRODUCIBILITY STATEMENT

We provide detailed materials to support the reproducibility of our results. Complete proofs of the theoretical claims, together with the corresponding assumptions and empirical measurements, are provided in Appendices A–B. Additional generalization and random-seed experiments are reported in Appendices E–F, while dynamic-reuse and runtime analyses are provided in Appendix G. Detailed training configurations, hyperparameters, and DGG implementation details are provided in Appendix I. Source code for reproducing our main results is included in the supplementary materials.

REFERENCES

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

Andrei Baroian and Rutger Berger. Prompt replay: speeding up grpo with on-policy reuse of high-signal prompts. *arXiv preprint arXiv:2603.21177*, 2026. URL https://arxiv.org/abs/2603.21177.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, et al. Skyrl-agent: Efficient rl training for multi-turn llm agent. *arXiv preprint arXiv:2511.16108*, 2025. URL https://arxiv.org/abs/2511.16108.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025. URL https://arxiv.org/abs/2506.13585.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng

Dou. Agentic reinforced policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TX4k7BF6aO.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=QXEhBMNrCW.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. URL https://www.nature.com/articles/s41586-025-09422-z.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

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. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, et al. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint arXiv:2508.17445*, 2025a. URL https://arxiv.org/abs/2508.17445.

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation. *arXiv preprint arXiv:2509.25849*, 2025b. URL https://arxiv.org/pdf/2509.25849.

Jing Liang, Jinyi Liu, Yi Ma, Hongyao Tang, YAN ZHENG, Shuyue Hu, LEI BAI, and Jianye HAO. Squeeze the soaked sponge: Efficient off-policy RFT for large language model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=quBjNSJMrC.

Mengqi Liao, Xiangyu Xi, Chen Ruinian, Jia Leng, Yangen Hu, Ke Zeng, Shuai Liu, and Huaiyu Wan. Enhancing efficiency and exploration in reinforcement learning for llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1451–1463, 2025. URL "https://aclanthology.org/2025.emnlp-main.75/".

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Shiqi Liu, Zeyu He, Guojian Zhan, Letian Tao, Zhilong Zheng, Jiang Wu, Yinuo Wang, Yang Guan, Kehua Sheng, Bo Zhang, et al. Stapo: Stabilizing reinforcement learning for llms by silencing rare spurious tokens. *arXiv preprint arXiv:2602.15620*, 2026. URL https://arxiv.org/abs/2602.15620.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2602.01601*, 2026. URL https://arxiv.org/abs/2602.01601.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015. URL https://arxiv.org/abs/1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5827–5837. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/simsekli19a.html.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017. URL https://arxiv.org/abs/1605.08254.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for LLM reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=uwUkETPIJN.

Meituan LongCat Team, Anchun Gui, Bei Li, Bingyang Tao, Bole Zhou, Borun Chen, Chao Zhang, Chengcheng Han, Chenhui Yang, Chi Zhang, et al. Introducing longcat-flash-thinking: A technical report. *arXiv preprint arXiv:2509.18883*, 2025. URL https://arxiv.org/abs/2509.18883.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. URL https://arxiv.org/abs/2302.13971.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 05 2022. ISSN 2307-387X. doi: 10.1162/tacl_a_00475. URL https://doi.org/10.1162/tacl_a_00475.

Chen Wang, Lai Wei, Yanzhi Zhang, Chenyang Shao, Zedong Dan, Weiran Huang, Yuzhi Zhang, and Yue Wang. Eframe: Deeper reasoning via exploration-filter-replay reinforcement learning framework. *arXiv preprint arXiv:2506.22200*, 2025a. URL https://arxiv.org/abs/2506.22200.

Jiakang Wang, Runze Liu, Lei Lin, Wenping Hu, Xiu Li, Fuzheng Zhang, Guorui Zhou, and Kun Gai. Aspo: Asymmetric importance sampling policy optimization. *arXiv preprint arXiv:2510.06062*, 2025b. URL https://arxiv.org/abs/2510.06062.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022. URL https://arxiv.org/abs/2212.03533.

Tao Wang and Sicun Gao. Numerical pitfalls in policy gradient updates. 2025. URL https://openreview.net/forum?id=u4dORXVAnx.

Ziyan Wang, Zheng Wang, Xingwei Qu, Qi Cheng, Jie Fu, Shengpu Tang, Minjia Zhang, and Xiaoming Huo. Slow-fast policy optimization: Reposition-before-update for llm reasoning. 2026. URL https://arxiv.org/abs/2510.04072.

Bo Wu, Sid Wang, Yunhao Tang, Jia Ding, Eryk Helenowski, Liang Tan, Tengyu Xu, Tushar Gowda, Zhengxing Chen, Chen Zhu, et al. Llamarl: A distributed asynchronous reinforcement learning framework for efficient large-scale llm training. *arXiv preprint arXiv:2505.24034*, 2025. URL https://arxiv.org/abs/2505.24034.

Shangyu Xing, Siyuan Wang, Chenyuan Yang, Xinyu Dai, and Xiang Ren. Lookahead tree-based rollouts for enhanced trajectory-level exploration in reinforcement learning with verifiable rewards. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4nLvUk8edu.

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

Kun Yang, Yanmeng Wang, Zhigen Li, et al. Sspo: Subsentence-level policy optimization. *arXiv preprint arXiv:2511.04256*, 2025b. URL https://arxiv.org/abs/2511.04256.

Zhihe Yang, Xufang Luo, Zilong Wang, Dongqi Han, Zhiyuan He, Dongsheng Li, and Yunjian Xu. Do not let low-probability tokens over-dominate in RL for LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FOnAdLo0tM.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 6672–6679, 2020. URL https://arxiv.org/abs/1912.09729.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017. URL https://arxiv.org/abs/1705.10941.

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. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. ExGRPO: Learning to reason from experience. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=701tjQXWVk.

Hao Zhang, Mingjie Liu, Shaokun Zhang, Songyang Han, Jian Hu, Zhenghui Jin, Yuchi Zhang, Shizhe Diao, Ximing Lu, Binfeng Xu, et al. Prorl agent: Rollout-as-a-service for rl training of multi-turn llm agents. *arXiv preprint arXiv:2603.18815*, 2026a. URL https://arxiv.org/abs/2603.18815.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15383–15393. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b05b57f6add810d3b7490866d74c0053-Paper.pdf.

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

Zhi Zhang, Zhen Han, Costas Mavromatis, Qi Zhu, Yunyi Zhang, Sheng Guan, Dingmin Wang, Xiong Zhou, Shuai Wang, Soji Adeshina, et al. Train less, learn more: Adaptive efficient rollout optimization for group-based reinforcement learning. *arXiv preprint arXiv:2602.14338*, 2026b. URL https://arxiv.org/abs/2602.14338.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 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 *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=IIgl5MWelz.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026b. URL https://openreview.net/forum?id=x5lITYXmW2.

A Detailed Proofs and Empirical Measurements for Section 4.2

This appendix provides detailed proofs for all theoretical results presented in Section 4.2, alongside empirical measurements supporting Theorem 1. To make this section self-contained, we restate each result prior to its proof. Throughout the appendix, we retain the notation introduced in the main text (see "Setup and notation" in Section 4.2).

A.1 Proof of Proposition 1 (Gradient Decomposition)

Proposition 2 (Restatement of Proposition 1)**.** *Let $\mathcal{L}_i$ denote the objective for token a_i, and define the error signal as $E_i \triangleq r_i \hat{A}_i (e_{a_i} - \pi_\theta(\cdot \,|h_{L,i})) \in \mathbb{R}^{d_{vocab}}$. The* `lm_head` *gradient and the gradient contribution through any intermediate linear layer at position i admit the following rank-1 forms:*

$$G_i^{lm} \triangleq E_i \, h_{L,i}^\top, \qquad G_i^{int} \triangleq \big(J_i^\top E_i\big)\big(x_i^{int}\big)^\top, \tag{9}$$

where $J_i = \partial z_i / \partial y_i$ is the Jacobian of logits z_i with respect to the intermediate output $y_i = W_{int} x_i^{int}$.

Proof. The proof relies on the standard multivariate chain rule applied to the computational graph of the transformer architecture. We decouple the derivation into two steps, tracking the gradient flow from the final loss down to the respective layer weights.

Recall that the policy distribution $\pi_\theta(\cdot \mid h_{L,i})$ is parameterized by the softmax function applied to the final pre-activation logits $z_i \in \mathbb{R}^{d_{\text{vocab}}}$. In the unclipped region of the GRPO surrogate objective, the objective for a single token a_i is given by $\mathcal{L}_i = r_i \hat{A}_i = \frac{\pi_\theta(a_i|h_{L,i})}{\pi_{\text{old}}(a_i|h_{L,i})} \hat{A}_i$.

To compute the gradient of $\mathcal{L}_i$ with respect to the logits z_i, we apply the chain rule:

$$\nabla_{z_i} \mathcal{L}_i = \frac{\partial \mathcal{L}_i}{\partial \pi_\theta(a_i \mid h_{L,i})} \nabla_{z_i} \pi_\theta(a_i \mid h_{L,i}). \tag{10}$$

The derivative of the loss with respect to the scalar probability of the sampled token is simply:

$$\frac{\partial \mathcal{L}_i}{\partial \pi_\theta(a_i \mid h_{L,i})} = \frac{\hat{A}_i}{\pi_{\text{old}}(a_i \mid h_{L,i})}. \tag{11}$$

Furthermore, the gradient of the softmax probability $\pi_\theta(a_i \mid h_{L,i})$ with respect to the full logit vector z_i evaluates to:

$$\nabla_{z_i} \pi_\theta(a_i \mid h_{L,i}) = \pi_\theta(a_i \mid h_{L,i}) \big(e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\big), \tag{12}$$

where $e_{a_i} \in \mathbb{R}^{d_{\text{vocab}}}$ is the one-hot indicator vector for the sampled token a_i. Multiplying these two components together, the behavior policy probability π_{old} and the current policy probability π_θ merge back to recover the importance ratio r_i:

$$\nabla_{z_i} \mathcal{L}_i = \frac{\pi_\theta(a_i \mid h_{L,i})}{\pi_{\text{old}}(a_i \mid h_{L,i})} \hat{A}_i \big(e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\big) = r_i \hat{A}_i \big(e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\big) \triangleq E_i. \tag{13}$$

The forward pass of the `lm_head` is a linear projection mapping the final hidden state $h_{L,i} \in \mathbb{R}^{d_{\text{model}}}$ to the logits $z_i \in \mathbb{R}^{d_{\text{vocab}}}$:

$$z_i = W_{\text{lm}} h_{L,i}. \tag{14}$$

To derive the gradient with respect to the weight matrix $W_{\text{lm}} \in \mathbb{R}^{d_{\text{vocab}} \times d_{\text{model}}}$, we apply the chain rule at the scalar level. For any specific entry $(W_{\text{lm}})_{k,m}$, it only contributes to the k-th logit $(z_i)_k = \sum_{m'} (W_{\text{lm}})_{k,m'} (h_{L,i})_{m'}$. Thus, the partial derivative is:

$$\frac{\partial \mathcal{L}_i}{\partial (W_{\text{lm}})_{k,m}} = \frac{\partial \mathcal{L}_i}{\partial (z_i)_k} \frac{\partial (z_i)_k}{\partial (W_{\text{lm}})_{k,m}} = (E_i)_k (h_{L,i})_m. \tag{15}$$

Reconstructing these element-wise partial derivatives into matrix form precisely yields the outer product of the upstream gradient vector E_i and the input vector $h_{L,i}$:

$$G_i^{\text{lm}} \triangleq \nabla_{W_{\text{lm}}} \mathcal{L}_i = E_i h_{L,i}^\top. \tag{16}$$

This establishes the first identity.

For an arbitrary intermediate linear layer, let its output at position i be $y_i = W_{\text{int}} x_i^{\text{int}}$, where x_i^{int} is the input to this layer and y_i is its pre-activation output. Let $J_i = \frac{\partial z_i}{\partial y_i}$ denote the composite Jacobian matrix, where its (k, m)-th entry is $(J_i)_{k,m} = \frac{\partial (z_i)_k}{\partial (y_i)_m}$. Since the off-policy anomaly originates from the token-level error E_i at position i, we analyze the position-aligned contribution induced by the application of W_{int} at the same position. This component is well-defined for every intermediate linear layer, while the composite Jacobian J_i absorbs all downstream transformations.

To find the upstream gradient arriving at the vector y_i, we apply the multivariate chain rule to its m-th component:

$$\frac{\partial \mathcal{L}_i}{\partial (y_i)_m} = \sum_{k=1}^{d_{\text{vocab}}} \frac{\partial \mathcal{L}_i}{\partial (z_i)_k} \frac{\partial (z_i)_k}{\partial (y_i)_m} = \sum_{k=1}^{d_{\text{vocab}}} (E_i)_k (J_i)_{k,m}. \tag{17}$$

Recognizing this sum as a matrix-vector product, the full upstream gradient vector can be compactly written using the transposed Jacobian:

$$\nabla_{y_i} \mathcal{L}_i = J_i^\top E_i. \tag{18}$$

Next, we consider the gradient contribution associated with this application of W_{int} at position i. Combining the upstream gradient $\nabla_{y_i} \mathcal{L}_i$ with the layer input x_i^{int}, its (m, p)-th entry is

$$(G_i^{\text{int}})_{m,p} = (\nabla_{y_i} \mathcal{L}_i)_m (x_i^{\text{int}})_p. \tag{19}$$

Assembling these entries into matrix form yields the outer product:

$$G_i^{\text{int}} \triangleq (\nabla_{y_i} \mathcal{L}_i)(x_i^{\text{int}})^\top = \left(J_i^\top E_i\right)\left(x_i^{\text{int}}\right)^\top. \tag{20}$$

For shared Transformer parameters, G_i^{int} denotes this position-aligned contribution associated with the application of W_{int} at position i. This establishes the second identity and completes the proof of Proposition 1. □

Remark on token-level analysis. A natural question is why Proposition 1 is stated at the token level. Our token-level formulation is a deliberate analytical choice driven by the nature of sample-reuse-induced collapse. As established in Section 3, this collapse is mechanistically driven by a small subset of *tail tokens*—those with vanishingly small $\pi_{\text{old}}(a_i)$ but massively inflated importance ratios r_i and large advantages, which dominate the gradient and trigger the anomaly. Proposition 1 is therefore designed to localize the structural asymmetry precisely on these harmful tokens: the policy-confidence factor $(1 - \pi_\theta(a_i))^{-2}$ is tightest exactly when $\pi_\theta(a_i)$ is small, faithfully capturing the regime where collapse originates. A batch-level statement, by averaging over many benign tokens, would dilute this signal and obscure the very mechanism we aim to expose. The empirical surge in $\|G^{\text{lm}}\|_F^2$ observed in Figures 1 and 3—which is itself a batch-level quantity—is consistent with the token-level analysis precisely because tail tokens dominate the gradient signal during collapse.

A.2 Proof of Lemma 1 (Bounded Activations under Pre-Norm Architectures)

Lemma 3 (Restatement of Lemma 1)**.** *For any active token i, let $h_{L,i}$ denote the* `lm_head` *input and x_i^{int} denote the intermediate-layer input. There exist strictly positive constants $\alpha_{\min}$ and $\beta_{\max}$ such that:*

$$\|h_{L,i}\|_2^2 \;\geq\; \alpha_{\min} \cdot d_{model}, \qquad \|x_i^{int}\|_2^2 \;\leq\; \beta_{\max} \cdot d_{model}. \tag{21}$$

Proof. We establish the two bounds separately, as they involve activations from distinct sub-modules: $h_{L,i}$ is produced by the final RMSNorm preceding the `lm_head`, whereas x_i^{int} is the input to an arbitrary intermediate linear layer.

Lower bound on $\|h_{L,i}\|_2^2$. The `lm_head` input takes the form $h_{L,i} = \text{RMSNorm}(v_L) \odot w^{\text{lm}}$, where $w^{\text{lm}} \in \mathbb{R}^{d_{\text{model}}}$ is the affine scale parameter of the final RMSNorm layer, and $\odot$ denotes element-wise multiplication. By the definition of the squared ℓ_2 norm and the RMSNorm operation, we have:

$$\|h_{L,i}\|_2^2 \;=\; \sum_{k=1}^{d_{\text{model}}} \frac{(v_L)_k^2}{\frac{1}{d_{\text{model}}}\|v_L\|_2^2 + \epsilon} (w_k^{\text{lm}})^2, \tag{22}$$

where ϵ is a fixed numerical stabilizer (typically 10^{-5}). To simplify the analysis, let $r \triangleq \frac{1}{d_{\text{model}}}\|v_L\|_2^2$ denote the mean squared magnitude of the pre-normalization vector v_L. The denominator in the summation can then be rewritten as $r + \epsilon$. Since this denominator is constant with respect to the summation index k, we can factor it out:

$$\|h_{L,i}\|_2^2 \;=\; \frac{1}{r+\epsilon}\sum_{k=1}^{d_{\text{model}}}(v_L)_k^2\,(w_k^{\text{lm}})^2. \tag{23}$$

Next, we multiply and divide the terms inside the summation by $\|v_L\|_2^2$ to construct a weighted average:

$$\|h_{L,i}\|_2^2 \;=\; \frac{1}{r+\epsilon}\sum_{k=1}^{d_{\text{model}}}\left(\frac{(v_L)_k^2}{\|v_L\|_2^2}\right)\cdot\|v_L\|_2^2\cdot(w_k^{\text{lm}})^2. \tag{24}$$

Factoring out $\|v_L\|_2^2$ (which is equal to $r \cdot d_{\text{model}}$ by definition) yields:

$$\|h_{L,i}\|_2^2 \;=\; \frac{r\cdot d_{\text{model}}}{r+\epsilon}\sum_{k=1}^{d_{\text{model}}}\frac{(v_L)_k^2}{\|v_L\|_2^2}\,(w_k^{\text{lm}})^2. \tag{25}$$

Observe that the coefficients $c_k = \frac{(v_L)_k^2}{\|v_L\|_2^2}$ satisfy $c_k \geq 0$ for all k and $\sum_k c_k = 1$. Therefore, the summation is a convex combination of the squared scale parameters $\{(w_k^{\text{lm}})^2\}$. Since a convex combination of a set of values is lower-bounded by their minimum, we obtain:

$$\|h_{L,i}\|_2^2 \;\geq\; \frac{r}{r+\epsilon}\cdot d_{\text{model}}\cdot\Big(\min_k(w_k^{\text{lm}})^2\Big). \tag{26}$$

Finally, we observe that the mean energy of the pre-normalization vector typically exceeds the numerical stabilizer by a large margin in trained Transformers. Specifically, with $\epsilon = 10^{-5}$ and $d_{\text{model}} \geq 10^3$, the condition $r \geq \epsilon$ merely requires $\|v_L\|_2^2 \geq 10^{-2}$, which is easily satisfied in practice. Consequently, the monotonically increasing function $f(r) = \frac{r}{r+\epsilon}$ safely satisfies $f(r) \geq f(\epsilon) = \frac{1}{2}$, yielding the final bound:

$$\|h_{L,i}\|_2^2 \;\geq\; \alpha_{\min}\cdot d_{\text{model}}, \qquad \text{where} \quad \alpha_{\min} \;\triangleq\; \frac{1}{2}\min_k(w_k^{\text{lm}})^2. \tag{27}$$

This establishes the strictly positive lower bound for the `lm_head` input.

Upper bound on $\|x_i^{\text{int}}\|_2^2$. We classify intermediate linear layers into two classes based on their input source and bound each class separately.

Class I: Inputs directly from RMSNorm. The inputs to QKV/up/gate projections take the form $x = \text{RMSNorm}(v)\odot w^{\text{int}}$, where w^{int} is the affine weight of an intermediate RMSNorm layer. Following the same factorization as above and using the convex-combination upper bound,

$$\|x\|_2^2 \;=\; \frac{r}{r+\epsilon}\cdot d_{\text{model}}\sum_k\frac{v_k^2}{\|v\|_2^2}(w_k^{\text{int}})^2 \;\leq\; \Big(\max_k(w_k^{\text{int}})^2\Big)\cdot d_{\text{model}}, \tag{28}$$

where the inequality uses $\frac{r}{r+\epsilon} \leq 1$. Defining $\beta^{\text{RMS}} \triangleq \max_k(w_k^{\text{int}})^2$ (taken over all intermediate RMSNorm layers), we have

$$\|x\|_2^2 \;\leq\; \beta^{\text{RMS}}\cdot d_{\text{model}}. \tag{29}$$

Class II: Inputs from attention or FFN internal states. We treat the two cases separately.

(a) Attention output projection W_O. The input is $x_i^O = \sum_j \alpha_{ij}v_j$, where $v_j = W_V x_j^{\text{RMS}}$ are value vectors and $\alpha_{ij} \geq 0$ with $\sum_j \alpha_{ij} = 1$. By Jensen's inequality applied to the convex function $\|\cdot\|_2^2$,

$$\|x_i^O\|_2^2 \;\leq\; \sum_j \alpha_{ij}\|v_j\|_2^2 \;\leq\; \max_j\|v_j\|_2^2. \tag{30}$$

Since $\|v_j\|_2^2 \leq \|W_V\|_2^2\cdot\|x_j^{\text{RMS}}\|_2^2 \leq \rho_V^2\beta^{\text{RMS}}d_{\text{model}}$ where $\rho_V \triangleq \|W_V\|_2$ is the operator norm implicitly controlled via weight decay, we obtain

$$\|x_i^O\|_2^2 \;\leq\; \rho_V^2\beta^{\text{RMS}}\cdot d_{\text{model}}. \tag{31}$$

(b) FFN down projection W_{down}. We treat the FFN down projection input in a manner agnostic to the specific activation function used. All standard activations σ in modern Transformers are Lipschitz continuous, satisfying $|\sigma(z)| \le L_\sigma |z|$ for some constant $L_\sigma > 0$ (e.g., $L_\sigma = 1$ for ReLU and SiLU/Swish, $L_\sigma \approx 1.13$ for GELU).

For gated variants (SwiGLU, GeGLU), the FFN down input is $x_i^{\text{down}} = \sigma(W_{\text{gate}} x_i^{\text{RMS}}) \odot (W_{\text{up}} x_i^{\text{RMS}})$. Applying the Hadamard product bound $\|u \odot v\|_2 \le \|u\|_\infty \|v\|_2$:

$$\|x_i^{\text{down}}\|_2 \;\le\; \|\sigma(W_{\text{gate}} x_i^{\text{RMS}})\|_\infty \cdot \|W_{\text{up}} x_i^{\text{RMS}}\|_2. \tag{32}$$

The first factor satisfies $\|\sigma(W_{\text{gate}} x_i^{\text{RMS}})\|_\infty \le L_\sigma \|W_{\text{gate}} x_i^{\text{RMS}}\|_\infty \le L_\sigma B_{\text{gate}}$, where $B_{\text{gate}} > 0$ is a finite constant bounding the pre-activation coordinate amplitudes—a property maintained in any trained network through the joint effect of the preceding RMSNorm and weight decay regularization on W_{gate}.

The second factor admits the standard spectral-norm bound $\|W_{\text{up}} x_i^{\text{RMS}}\|_2 \le \rho_{\text{up}} \sqrt{\beta^{\text{RMS}} d_{\text{model}}}$, where the operator norm $\rho_{\text{up}} \triangleq \|W_{\text{up}}\|_2$ is implicitly controlled via weight decay. Combining these terms, we obtain:

$$\|x_i^{\text{down}}\|_2^2 \;\le\; L_\sigma^2 B_{\text{gate}}^2 \rho_{\text{up}}^2 \beta^{\text{RMS}} \cdot d_{\text{model}} \;\triangleq\; \rho_{\text{FFN}}^2 \beta^{\text{RMS}} \cdot d_{\text{model}}. \tag{33}$$

For non-gated variants ($x_i^{\text{down}} = \sigma(W_{\text{up}} x_i^{\text{RMS}})$), the coordinate-wise Lipschitz property of σ implies $\|\sigma(z)\|_2 \le L_\sigma \|z\|_2$, and the operator-norm bound directly gives $\|x_i^{\text{down}}\|_2^2 \le L_\sigma^2 \rho_{\text{up}}^2 \beta^{\text{RMS}} d_{\text{model}}$, fitting the same form (subsumed by ρ_{FFN}^2 defined to cover both cases).

Unified upper bound. Define

$$\beta_{\max} \;\triangleq\; \max\!\big(\beta^{\text{RMS}},\; \rho_V^2 \beta^{\text{RMS}},\; \rho_{\text{FFN}}^2 \beta^{\text{RMS}}\big). \tag{34}$$

Combining the upper bounds across Class I and Class II yields $\|x_i^{\text{int}}\|_2^2 \le \beta_{\max} \cdot d_{\text{model}}$ uniformly across all intermediate linear layers. Note that $\alpha_{\min}$ and $\beta_{\max}$ are determined by separate sub-modules of the network—the final RMSNorm affine weight w^{lm} for $\alpha_{\min}$, and the intermediate RMSNorm affine weights w^{int} together with attention/FFN operator norms for $\beta_{\max}$. The empirical values of $\beta_{\max}/\alpha_{\min}$ are reported across diverse architectures in Appendix A.4. □

A.3 Proof of Theorem 1 (Structural Gradient Asymmetry)

Theorem 3 (Restatement of Theorem 1)**.** *Under Lemma 1 and Assumption 1, for any sampled token a_i with policy probability $\pi_\theta(a_i) < 1$ and non-zero advantage, we have:*

$$\frac{\|G_i^{int}\|_F^2}{\|G_i^{lm}\|_F^2} \;\le\; \mathcal{C}_{struct} \cdot \frac{1}{(1-\pi_\theta(a_i))^2}, \tag{35}$$

where $\mathcal{C}_{struct} \triangleq \frac{4\beta_{\max} C}{\alpha_{\min}}$ is a strictly positive architectural constant.

Proof. Throughout this proof, we suppress the conditioning of π_θ on $h_{L,i}$ for notational brevity, writing $\pi_\theta(j)$ for $\pi_\theta(j \mid h_{L,i})$.

We first establish a useful identity for the Frobenius norm of rank-1 matrices. For any vectors $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$, the outer product $M = uv^\top \in \mathbb{R}^{n \times m}$ has entries $M_{j,k} = u_j v_k$. By definition of the Frobenius norm and applying the distributive law to the double sum,

$$\|M\|_F^2 = \sum_{j=1}^{n}\sum_{k=1}^{m} M_{j,k}^2 = \sum_{j=1}^{n}\sum_{k=1}^{m} u_j^2 v_k^2 = \left(\sum_{j=1}^{n} u_j^2\right)\left(\sum_{k=1}^{m} v_k^2\right) = \|u\|_2^2 \cdot \|v\|_2^2. \tag{36}$$

We apply this identity to both G_i^{lm} and G_i^{int}, both of which are rank-1 outer products by Proposition 1.

Step 1: Lower bound on $\|G_i^{\text{lm}}\|_F^2$. By Proposition 1, $G_i^{\text{lm}} = E_i h_{L,i}^\top$. Applying Eq. 36,

$$\|G_i^{\text{lm}}\|_F^2 = \|E_i\|_2^2 \cdot \|h_{L,i}\|_2^2. \tag{37}$$

We bound $\|E_i\|_2^2$ from below. Recall $E_i = r_i \hat{A}_i (e_{a_i} - \pi_\theta)$, whose entries are

$$(E_i)_j = \begin{cases} r_i \hat{A}_i \big(1 - \pi_\theta(a_i)\big), & j = a_i, \\ -r_i \hat{A}_i \pi_\theta(j), & j \ne a_i. \end{cases} \tag{38}$$

By definition of the squared ℓ_2 norm and isolating the $j = a_i$ component,

$$\|E_i\|_2^2 = \sum_{j=1}^{d_{\text{vocab}}} (E_i)_j^2 \tag{39}$$

$$= (E_i)_{a_i}^2 + \sum_{j \neq a_i} (E_i)_j^2 \tag{40}$$

$$= r_i^2 \hat{A}_i^2 \left[\big(1 - \pi_\theta(a_i)\big)^2 + \sum_{j \neq a_i} \pi_\theta(j)^2 \right] \tag{41}$$

$$\geq r_i^2 \hat{A}_i^2 \cdot \big(1 - \pi_\theta(a_i)\big)^2, \tag{42}$$

where the final inequality holds since $\sum_{j \neq a_i} \pi_\theta(j)^2 \geq 0$ (each term is non-negative). Combining Eq. 42 with Lemma 1's lower bound $\|h_{L,i}\|_2^2 \geq \alpha_{\min} d_{\text{model}}$,

$$\|G_i^{\text{lm}}\|_F^2 \;\geq\; r_i^2 \hat{A}_i^2 \cdot \big(1 - \pi_\theta(a_i)\big)^2 \cdot \alpha_{\min} \cdot d_{\text{model}}. \tag{43}$$

Step 2: Upper bound on $\|G_i^{\text{int}}\|_F^2$. By Proposition 1, $G_i^{\text{int}} = (J_i^\top E_i)(x_i^{\text{int}})^\top$. Applying Eq. 36,

$$\|G_i^{\text{int}}\|_F^2 = \|J_i^\top E_i\|_2^2 \cdot \|x_i^{\text{int}}\|_2^2. \tag{44}$$

We focus on bounding $\|J_i^\top E_i\|_2^2 = \sum_{k=1}^{d_{\text{model}}} (J_i^\top E_i)_k^2$.

Closed-form for $(J_i^\top E_i)_k$. By the matrix-vector product definition and substituting the entries of E_i:

$$(J_i^\top E_i)_k = \sum_{j=1}^{d_{\text{vocab}}} (J_i)_{j,k} (E_i)_j \qquad \text{(matrix-vector definition)}$$

$$= r_i \hat{A}_i \left[(J_i)_{a_i,k} \big(1 - \pi_\theta(a_i)\big) - \sum_{j \neq a_i} (J_i)_{j,k} \pi_\theta(j) \right] \qquad \text{(substituting } (E_i)_j\text{)}$$

$$= r_i \hat{A}_i \left[(J_i)_{a_i,k} - \pi_\theta(a_i)(J_i)_{a_i,k} - \sum_{j \neq a_i} \pi_\theta(j)(J_i)_{j,k} \right] \qquad \text{(expanding the first product)}$$

$$= r_i \hat{A}_i \left[(J_i)_{a_i,k} - \sum_{j=1}^{d_{\text{vocab}}} \pi_\theta(j)(J_i)_{j,k} \right] \qquad \text{(merging into a full sum over } j\text{)}$$

$$= r_i \hat{A}_i \Big((J_i)_{a_i,k} - \mathbb{E}_{j \sim \pi_\theta} \big[(J_i)_{j,k} \big] \Big). \tag{45}$$

Squaring and applying an elementary inequality. For any real numbers a, b, expanding $(a-b)^2 = a^2 - 2ab + b^2$ and applying the AM-GM inequality $-2ab \leq a^2 + b^2$ yields $(a-b)^2 \leq 2a^2 + 2b^2$. Applying this to Eq. 45,

$$\big((J_i^\top E_i)_k\big)^2 \;\leq\; 2 r_i^2 \hat{A}_i^2 \Big[(J_i)_{a_i,k}^2 + \big(\mathbb{E}_{j \sim \pi_\theta} [(J_i)_{j,k}] \big)^2 \Big]. \tag{46}$$

Summing over k and decomposing. Summing Eq. 46 over $k = 1, \ldots, d_{\text{model}}$,

$$\|J_i^\top E_i\|_2^2 = \sum_{k=1}^{d_{\text{model}}} \big((J_i^\top E_i)_k\big)^2 \;\leq\; 2 r_i^2 \hat{A}_i^2 \Bigg[\underbrace{\sum_{k=1}^{d_{\text{model}}} (J_i)_{a_i,k}^2}_{\text{Term I}} + \underbrace{\sum_{k=1}^{d_{\text{model}}} \big(\mathbb{E}_{j \sim \pi_\theta} [(J_i)_{j,k}] \big)^2}_{\text{Term II}} \Bigg]. \tag{47}$$

We bound each term using Assumption 1.

Bound on Term I. By definition of the row ℓ_2 norm,

$$\text{Term I} = \sum_{k=1}^{d_{\text{model}}} (J_i)_{a_i,k}^2 = \|(J_i)_{a_i,:}\|_2^2 \;\leq\; C, \tag{48}$$

where the inequality applies the sampled-token bound in Assumption 1(ii).

Bound on Term II. Applying Jensen's inequality coordinate-wise (the function $x \mapsto x^2$ is convex on $\mathbb{R}$),

$$\left(\mathbb{E}_{j\sim\pi_\theta}[(J_i)_{j,k}]\right)^2 \le \mathbb{E}_{j\sim\pi_\theta}\left[(J_i)^2_{j,k}\right], \quad \forall k. \tag{49}$$

Summing Eq. 49 over k and exchanging the finite sum with the expectation by linearity,

$$\begin{aligned}
\text{Term II} &\le \sum_{k=1}^{d_{\text{model}}} \mathbb{E}_{j\sim\pi_\theta}\left[(J_i)^2_{j,k}\right] && \text{(by Eq. 49)} \\
&= \mathbb{E}_{j\sim\pi_\theta}\left[\sum_{k=1}^{d_{\text{model}}} (J_i)^2_{j,k}\right] && \text{(linearity of expectation; finite sum)} \\
&= \mathbb{E}_{j\sim\pi_\theta}\left[\|(J_i)_{j,:}\|_2^2\right] && \text{(definition of row } \ell_2 \text{ norm)} \\
&\le C, && (50)
\end{aligned}$$

where the final inequality applies the expected logit sensitivity bound in Assumption 1(i).

Combining Term I and Term II. Substituting Eq. 48 and Eq. 50 into Eq. 47,

$$\|J_i^\top E_i\|_2^2 \le 2r_i^2\hat{A}_i^2 \cdot (C + C) = 4C \cdot r_i^2\hat{A}_i^2. \tag{51}$$

Substituting Eq. 51 into Eq. 44 and applying Lemma 1's upper bound $\|x_i^{\text{int}}\|_2^2 \le \beta_{\max} d_{\text{model}}$,

$$\|G_i^{\text{int}}\|_F^2 \le 4C \cdot r_i^2\hat{A}_i^2 \cdot \beta_{\max} \cdot d_{\text{model}} = 4\beta_{\max}C \cdot r_i^2\hat{A}_i^2 \cdot d_{\text{model}}. \tag{52}$$

Step 3: Combining the bounds. The hypothesis $\pi_\theta(a_i) < 1$ ensures $(1 - \pi_\theta(a_i))^2 > 0$, so $\frac{1}{\alpha_{\min}(1-\pi_\theta(a_i))^2}$ is well-defined and strictly positive. Multiplying Eq. 43 by $\frac{4\beta_{\max}C/\alpha_{\min}}{(1-\pi_\theta(a_i))^2}$ on both sides yields

$$\frac{4\beta_{\max}C}{\alpha_{\min}} \cdot \frac{1}{(1-\pi_\theta(a_i))^2} \cdot \|G_i^{\text{lm}}\|_F^2 \ge 4\beta_{\max}C \cdot r_i^2\hat{A}_i^2 \cdot d_{\text{model}}. \tag{53}$$

Combining with Eq. 52,

$$\|G_i^{\text{int}}\|_F^2 \le \frac{4\beta_{\max}C}{\alpha_{\min}} \cdot \frac{1}{(1-\pi_\theta(a_i))^2} \cdot \|G_i^{\text{lm}}\|_F^2. \tag{54}$$

Under the hypothesis $\hat{A}_i \neq 0$, we have $\|G_i^{\text{lm}}\|_F^2 > 0$ (since Eq. 43 gives a strictly positive lower bound when $\hat{A}_i \neq 0$, $r_i > 0$, and $\pi_\theta(a_i) < 1$). Dividing both sides of Eq. 54 by $\|G_i^{\text{lm}}\|_F^2$,

$$\frac{\|G_i^{\text{int}}\|_F^2}{\|G_i^{\text{lm}}\|_F^2} \le \frac{4\beta_{\max}C}{\alpha_{\min}} \cdot \frac{1}{(1-\pi_\theta(a_i))^2}. \tag{55}$$

Identifying $\mathcal{C}_{\text{struct}} \triangleq 4\beta_{\max}C/\alpha_{\min}$ completes the proof. □

A.4 Empirical Measurements of Architectural Constants

The quantitative instantiation in Section 4.2.1 relies on empirical estimates of the quantities underlying the bound in Theorem 1. This appendix details the measurement protocol and reports the resulting $\mathcal{C}_{\text{struct}} = 4\beta_{\max}C/\alpha_{\min}$ across ten mainstream LLMs spanning the Llama-3, Qwen-2.5, Qwen-3, and GPT-OSS families (1.5B–235B parameters, including both dense and MoE architectures). The underlying activation and Jacobian quantities are estimated from forward-pass activations and Jacobian measurements on a held-out validation set. Given the heavy-tailed distribution of gradient-related quantities in deep networks (Simsekli et al., 2019; Zhang et al., 2020), we report the median (Med.) and 95th percentile of the resulting $\mathcal{C}_{\text{struct}}$.

The empirical measurements reveal a consistent pattern across all tested architectures. The architectural constant $\mathcal{C}_{\text{struct}}$ takes small values at the median in every dense model, ranging from 4.9×10^{-3} (Qwen3-4B-Instuct-2507) to 6.1×10^{-2} (Qwen3-32B), and is even smaller for the MoE architecture Qwen3-235B-A22B (1.9×10^{-6}). Consequently, $\mathcal{C}_{\text{struct}}$—which serves as the prefactor of the bound on $\|G_i^{\text{int}}\|_F^2/\|G_i^{\text{lm}}\|_F^2$ given by Eq. 35—remains below 10^{-1} at the median across all models, and

below 0.5 even at the 95th-percentile across all tested models. These measurements quantitatively support the small architectural prefactor in Theorem 1, consistent with the localized `lm_head` divergence observed empirically in Section 4.1.

Stability of $\mathcal{C}_{\text{struct}}$ throughout RL training. A natural concern is whether $\mathcal{C}_{\text{struct}}$ remains small throughout RL training. Figure 6 tracks its 95th-percentile across three regimes: it stays well below 1 and exhibits no surge even at the Naive Reuse collapse onset, showing that the empirical structural factor remains stable throughout training and continues to support the attenuation characterized by Theorem 1.

Table 2: **Empirical measurements** of the Jacobian row-energy term C in Assumption 1 and the architectural factor $\mathcal{C}_{\text{struct}}$ in Theorem 1 across ten LLM architectures. We report the median (Med.) and 95th percentile (95%). $\mathcal{C}_{\text{struct}}$ remains below 10^{-1} at the median and below 0.5 at the 95th percentile across all tested models.

Model	C		$\mathcal{C}_{\text{struct}}$	
	Med.	95%	Med.	95%
Llama-3.2-3B-Inst.	8.39×10^{-2}	9.15×10^{-1}	1.26×10^{-2}	3.45×10^{-1}
Llama-3.1-8B-Inst.	1.50×10^{-1}	1.97	1.49×10^{-2}	2.11×10^{-1}
Qwen2.5-1.5B-Inst.	1.64×10^{-1}	6.45×10^{-1}	1.29×10^{-2}	8.90×10^{-2}
Qwen2.5-7B-Inst.	3.12×10^{-2}	9.36×10^{-2}	1.68×10^{-2}	1.86×10^{-1}
Qwen2.5-72B-Inst.	1.07×10^{-1}	5.03×10^{-1}	5.26×10^{-3}	5.92×10^{-2}
Qwen3-4B-Inst-2507	7.56×10^{-2}	3.84×10^{-1}	4.93×10^{-3}	2.35×10^{-1}
Qwen3-8B	2.45×10^{-1}	8.04×10^{-1}	1.77×10^{-2}	3.67×10^{-1}
Qwen3-32B	7.61×10^{-1}	2.85	6.06×10^{-2}	1.74×10^{-1}
Qwen3-235B-A22B	2.83×10^{-5}	1.58×10^{-3}	1.85×10^{-6}	3.48×10^{-4}
GPT-OSS-120B	2.91×10^{-3}	4.86×10^{-3}	3.21×10^{-3}	1.03×10^{-2}

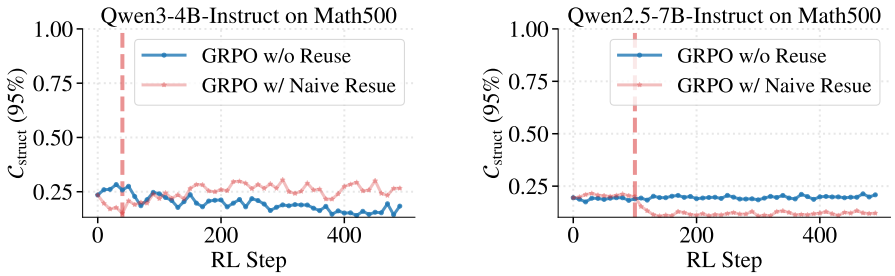

Figure 6: **Dynamics of $\mathcal{C}_{\text{struct}}$ (95th-percentile) during RL training** on Qwen3-4B-Instruct and Qwen2.5-7B-Instruct (Math500). Red dashed lines mark the onset of performance degradation under Naive Reuse. Observation: *$\mathcal{C}_{\text{struct}}$ remains stable across all regimes, including at collapse onset.*

B Detailed Proofs for Section 4.2.2

This appendix provides detailed proofs for all theoretical results presented in Section 4.2.2. To make this section self-contained, we restate each result prior to its proof. Throughout the appendix, we retain the notation introduced in the main text

B.1 Proof of Lemma 2 (Gradient Norm in Terms of Importance-Ratio Second Moment)

Lemma 4 (Restatement of Lemma 2)**.** *Let $\overline{r^2} \triangleq \frac{1}{T}\sum_{i=1}^{T} r_i^2$ denote the empirical second moment of importance ratios. Then*

$$\|G^{lm}\|_F^2 \;\le\; c_{\max}\,\overline{r^2}, \qquad \textit{where} \quad c_{\max} \triangleq \max_{1\le i\le T} \hat{A}_i^2\,\|e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\|_2^2\,\|h_{L,i}\|_2^2. \tag{56}$$

Proof. The proof proceeds in three steps: (1) decomposing the squared gradient norm via Jensen's inequality, (2) factoring per-token contributions into rank-1 norms, and (3) absorbing the representation-dependent factors into $c_{\max}$.

Step 1: Jensen's inequality on the squared norm. By Proposition 1, the batch-level `lm_head` gradient admits the rank-1 sum decomposition:

$$G^{\text{lm}} = \frac{1}{T}\sum_{i=1}^{T} E_i\, h_{L,i}^{\top}, \qquad E_i = r_i \hat{A}_i\big(e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\big). \tag{57}$$

The squared Frobenius norm $\|\cdot\|_F^2$ is convex on matrix space. Applying Jensen's inequality with the uniform distribution over $\{1, \ldots, T\}$:

$$\|G^{\text{lm}}\|_F^2 = \left\| \frac{1}{T}\sum_{i=1}^{T} E_i h_{L,i}^{\top} \right\|_F^2 \;\le\; \frac{1}{T}\sum_{i=1}^{T} \|E_i h_{L,i}^{\top}\|_F^2. \tag{58}$$

Step 2: Rank-1 norm factorization. For any vectors $u \in \mathbb{R}^m, v \in \mathbb{R}^n$, the outer product satisfies $\|uv^{\top}\|_F^2 = \|u\|_2^2 \|v\|_2^2$. Applying this identity to each rank-1 term in Eq. 58:

$$\|E_i h_{L,i}^{\top}\|_F^2 = \|E_i\|_2^2 \cdot \|h_{L,i}\|_2^2. \tag{59}$$

Substituting the explicit form of E_i:

$$\|E_i\|_2^2 = r_i^2 \hat{A}_i^2 \|e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\|_2^2. \tag{60}$$

Combining these:

$$\|G^{\text{lm}}\|_F^2 \;\le\; \frac{1}{T}\sum_{i=1}^{T} r_i^2 \hat{A}_i^2 \|e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\|_2^2 \|h_{L,i}\|_2^2. \tag{61}$$

Step 3: Absorbing representation factors into $c_{\max}$. By the definition of $c_{\max}$, the per-token factor involving $\hat{A}_i^2$, $\|e_{a_i} - \pi_\theta\|^2$, and $\|h_{L,i}\|^2$ is bounded by $c_{\max}$ uniformly across $i \in \{1, \ldots, T\}$:

$$\hat{A}_i^2 \|e_{a_i} - \pi_\theta(\cdot \mid h_{L,i})\|_2^2 \|h_{L,i}\|_2^2 \;\le\; c_{\max}, \quad \forall i. \tag{62}$$

Substituting into Eq. 61:

$$\|G^{\text{lm}}\|_F^2 \;\le\; \frac{c_{\max}}{T}\sum_{i=1}^{T} r_i^2 = c_{\max} \cdot \overline{r^2}, \tag{63}$$

which establishes the claimed bound. □

B.2 Proof of Theorem 2 (`lm_head` Gradient Energy Lower-Bounds Policy Divergence)

Theorem 4 (Restatement of Theorem 2)**.** *Let $\hat{\chi}^2 \triangleq \frac{1}{T}\sum_{i=1}^{T}(r_i^2 - 1)$ denote the empirical estimator of the Pearson χ^2 divergence $\chi^2(\pi_\theta \| \pi_{old})$. Then*

$$\|G^{lm}\|_F^2 \;\le\; c_{\max}\big(1 + \hat{\chi}^2\big), \tag{64}$$

or equivalently, $\hat{\chi}^2 \ge \|G^{lm}\|_F^2 / c_{\max} - 1$.

Proof. The proof reduces to a direct algebraic decomposition of the empirical second moment $\overline{r^2}$. By the definition of $\hat{\chi}^2$:

$$\hat{\chi}^2 = \frac{1}{T}\sum_{i=1}^{T}(r_i^2 - 1) = \frac{1}{T}\sum_{i=1}^{T} r_i^2 - 1 = \overline{r^2} - 1, \tag{65}$$

which gives the identity $\overline{r^2} = 1 + \hat{\chi}^2$. Substituting this into the bound from Lemma 2:

$$\|G^{\text{lm}}\|_F^2 \;\le\; c_{\max}\overline{r^2} = c_{\max}(1 + \hat{\chi}^2), \tag{66}$$

which establishes Eq. 64. Rearranging this inequality (well-defined since $c_{\max} > 0$):

$$\hat{\chi}^2 \;\ge\; \frac{\|G^{\text{lm}}\|_F^2}{c_{\max}} - 1, \tag{67}$$

which establishes the equivalent lower-bound form. The two forms are mathematically equivalent. □

Remark on the empirical χ^2 statistic. The quantity $\hat{\chi}^2 = \frac{1}{T}\sum_i (r_i^2 - 1)$ is the natural batch-level analog of the Pearson χ^2 divergence: under the relation $\mathbb{E}_{a\sim\pi_{\text{old}}}[r^2] - 1 = \chi^2(\pi_\theta \| \pi_{\text{old}})$ for the population divergence, $\hat{\chi}^2$ is the empirical estimator obtained by replacing the population expectation with the batch sample mean. We work with $\hat{\chi}^2$ throughout, since the bound in Theorem 2 holds deterministically at the batch level without invoking any concentration argument.

C Pseudocode of Dynamic Gradient Gating (DGG)

We provide the full pseudocode of GRPO with Dynamic Gradient Gating (DGG) in Algorithm 1. DGG integrates as a lightweight wrapper around the standard sample-reuse loop, requiring only a single Frobenius-norm computation per step and no modification to the underlying objective or optimizer. For clarity, Algorithm 1 omits the initialization warm-up; in implementation, gating is enabled only after the running-statistics window is fully populated, as detailed in Appendix I.2.

Algorithm 1 GRPO with Dynamic Gradient Gating (DGG)

Require: Initial policy π_θ, reuse factor K, threshold τ

for each training iteration **do**
  Generate rollout batch $\mathcal{D}$ from $\pi_{\text{old}} \leftarrow \pi_\theta$
  for $k = 1, \ldots, K$ **do**
    Compute GRPO loss $\mathcal{L}_{\text{GRPO}}$ on $\mathcal{D}$ via Eq. 1 and backpropagate
    $g_t \leftarrow \|G^{\text{lm}}\|_F^2$; $\Delta g_t \leftarrow g_t - g_{\text{ref}}$ # g_{ref}: most recent accepted state
    $z_t \leftarrow (\Delta g_t - \mu_t)/(\sigma_t + \varepsilon)$ # tested against historical stats
    if $k > 1$ **and** $z_t > \tau$ **then**
      Zero all gradients; **break** # discard; history unchanged
    end if
    Step the Adam optimizer
    Update running statistics $(\mu_{t+1}, \sigma_{t+1})$ with Δg_t
    $g_{\text{ref}} \leftarrow g_t$ # update reference only after acceptance
  end for
end for

D Empirical Validation of DWD Phenomenon on Additional LLMs

In this section, we extend our profiling of DWD phenomenon in Section 4.1 to additional LLMs and tasks. Figure 7 reports the layer-wise weight dynamics of Qwen2.5-7B-Instruct across all four tasks, and Figure 8 reports the dynamics on mathematical reasoning across four additional LLMs from the Qwen-2.5, Qwen-3, and Llama-3 families (1.5B–14B). In all settings, sample reuse accelerates early convergence but is followed by severe performance degradation, synchronized with an abrupt, isolated surge in the `lm_head`, while MLP and Attention layers remain stable. These results confirm DWD as a broadly present phenomenon across diverse architectures, scales, and tasks, supporting our theoretical analysis in Section 4.2 and the design of DGG.

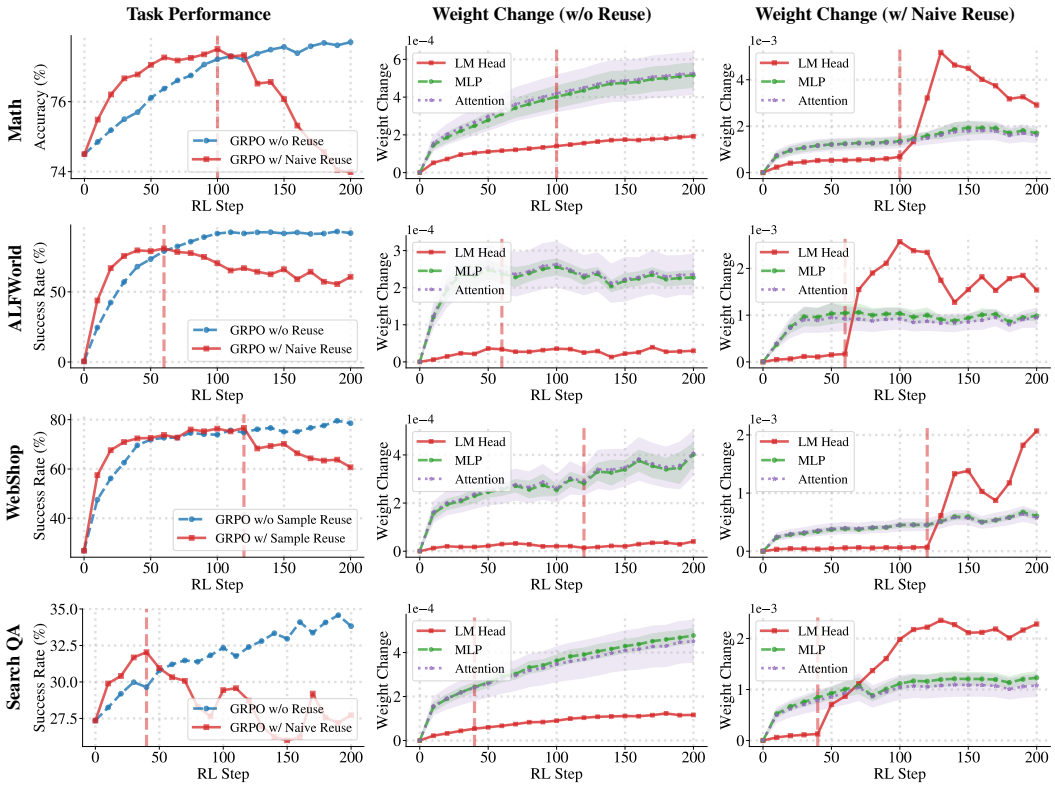

Figure 7: **Illustration of the Disproportionate Weight Divergence (DWD) phenomenon** on Qwen2.5-7B-Instruct across four tasks. Relative weight change is defined as $\|W_t - W_{t-\Delta}\|_F / \|W_{\text{ref}}\|_F$ (W_{ref}: initial pretrained weight; Δ: profiling interval, set to 10 RL steps); red dashed lines mark the onset of performance degradation for GRPO w/ Naive Reuse. Observation: *Sample reuse accelerates early convergence, followed by a severe performance degradation synchronized with an abrupt, isolated surge in the LM Head — while intermediate layers (Attention and MLP) remain stable.*

E Additional Generalization Results for DGG

We further evaluate DGG along three generalization axes not jointly covered by the main experiments: model family, policy-optimization algorithm, and model scale. Specifically, we evaluate Llama-3.2-3B-Instruct with GRPO on mathematical reasoning and search-augmented QA tasks, and Qwen2.5-7B/14B-Instruct with DAPO on mathematical reasoning task. These settings extend the main evaluation from Qwen-based GRPO training to a different model family, the DAPO training recipe, and models up to 14B parameters.

E.1 DGG under GRPO: Cross-Family and Cross-Domain Generalization

We first evaluate DGG on Llama-3.2-3B-Instruct under GRPO across mathematical reasoning and search-augmented QA. This setting examines whether DGG transfers beyond the Qwen family used in the main experiments and remains effective across distinct task domains.

As shown in Figure 9, DGG consistently stabilizes sample reuse across both domains. Naive reuse accelerates early optimization but subsequently exhibits instability, whereas DGG retains this acceleration while maintaining stable convergence and performance comparable to or better than single-use GRPO. Together with the Qwen results in the main text, these findings demonstrate that DGG is robust across model families and task domains.

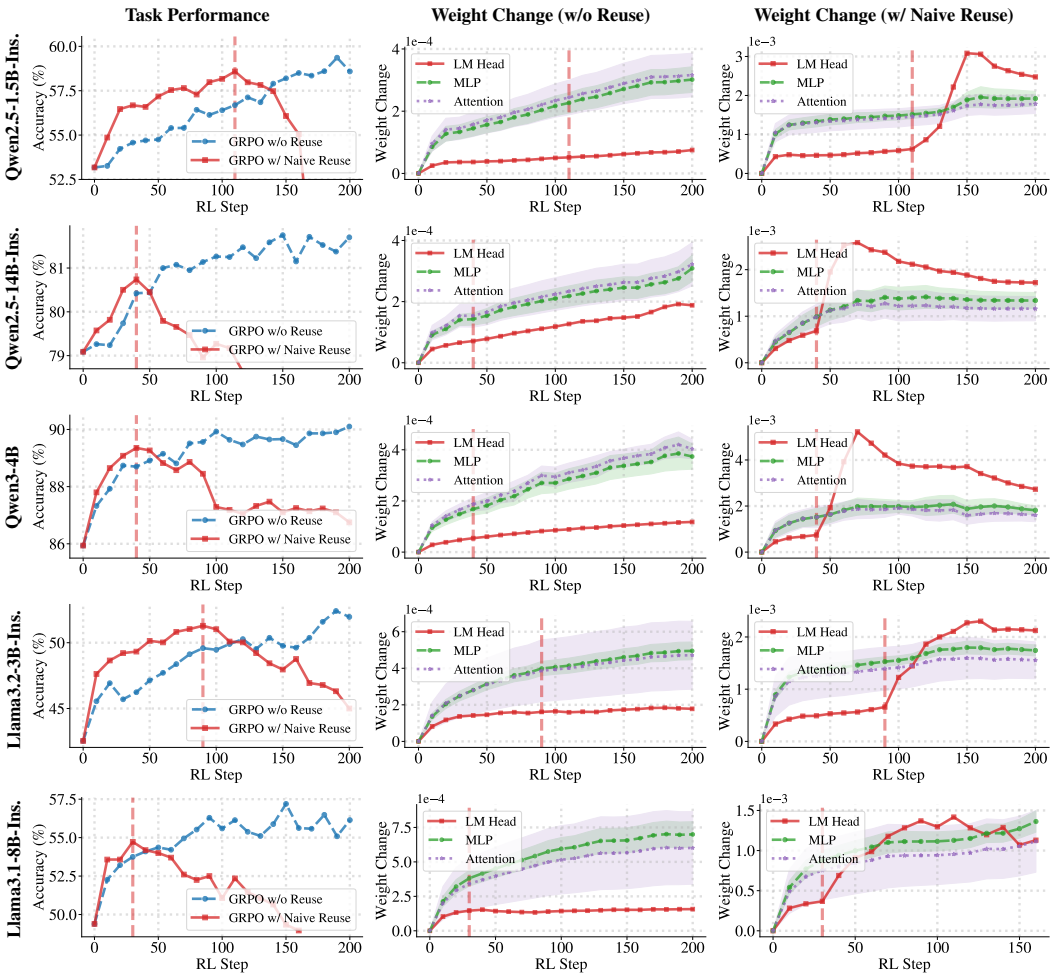

Figure 8: **Illustration of the Disproportionate Weight Divergence (DWD) phenomenon** on math task across different LLMs. Relative weight change is defined as $\|W_t - W_{t-\Delta}\|_F / \|W_{\text{ref}}\|_F$ (W_{ref}: initial pretrained weight; Δ: profiling interval, set to 10 RL steps); red dashed lines mark the onset of performance degradation for GRPO w/ Naive Reuse. Observation: *Sample reuse accelerates early convergence, followed by a severe performance degradation synchronized with an abrupt, isolated surge in the LM Head — while intermediate layers (Attention and MLP) remain stable.*

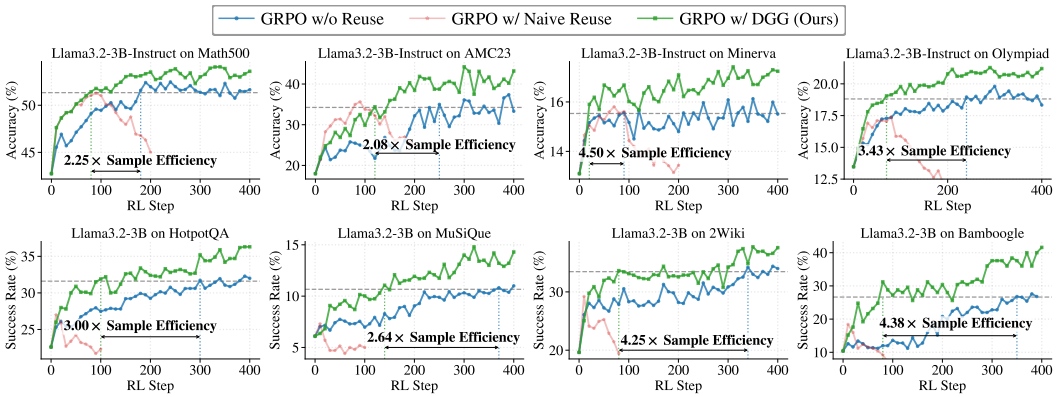

Figure 9: **Cross-family and cross-domain generalization of DGG under GRPO.** Each RL step corresponds to one fresh rollout batch; thus, methods at the same step are compared under **matched rollout budgets**. Results on Llama-3.2-3B-Instruct cover mathematical reasoning (top) and search-augmented QA (bottom). Observation: *DGG transfers beyond the Qwen family and consistently stabilizes sample reuse across both task domains, preserving its early acceleration while avoiding the instability of naive reuse.*

E.2 DGG UNDER DAPO: CROSS-SCALE GENERALIZATION

We next evaluate DGG under DAPO at the 7B and 14B scales, testing whether its gating mechanism transfers beyond GRPO and remains effective as model size increases.

As shown in Figure 10, DGG consistently stabilizes sample reuse across both scales and all four mathematical reasoning benchmarks. Naive reuse accelerates early optimization but subsequently becomes unstable, whereas DGG preserves this acceleration while maintaining stable convergence without performance degradation. Together with the GRPO results in the main text, these findings demonstrate that DGG generalizes across policy-optimization algorithms and scales from 7B to 14B.

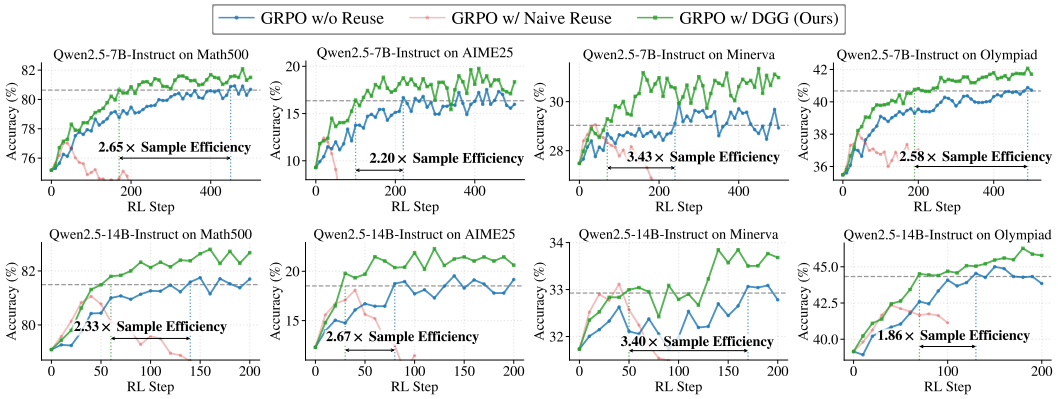

Figure 10: **Cross-scale generalization of DGG under DAPO.** Each RL step corresponds to one fresh rollout batch; thus, methods at the same step are compared under **matched rollout budgets**. Observation: *DGG remains effective when extending from GRPO to DAPO and from 7B to 14B, preserving the early acceleration of sample reuse while avoiding the instability of naive fixed reuse.*

F Stability of DGG across Random Seeds

To verify that DGG's improvements are robust to randomness rather than artifacts of a specific run, we repeat the Math500 experiments on Qwen3-4B-Instruct and Qwen2.5-7B-Instruct with three different random seeds. Figure 11 reports the mean accuracy with shaded standard deviation across seeds. DGG consistently outperforms the single-use baseline and avoids the collapse of Naive Reuse on every seed, with tight standard-deviation bands throughout training—confirming that DGG's gains are stable and reproducible.

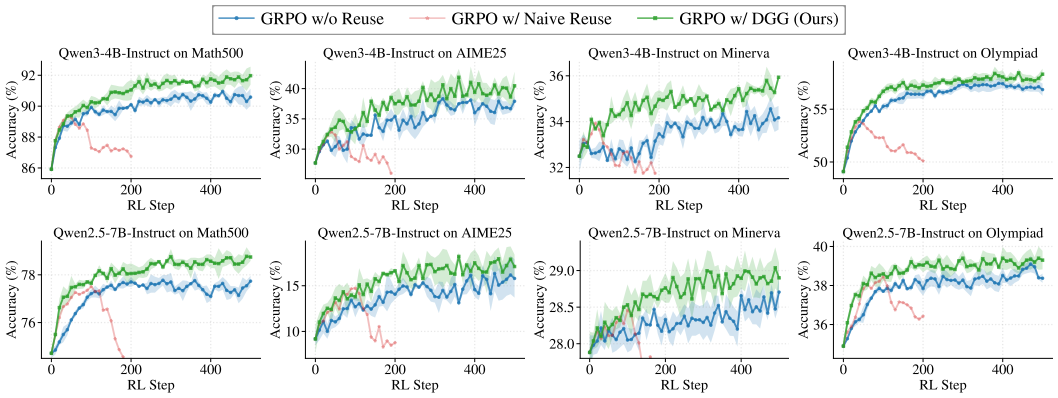

Figure 11: **Stability of DGG across three random seeds** for Qwen3-4B-Instruct and Qwen2.5-7B-Instruct on multiple benchmarks. Each RL step corresponds to one fresh rollout batch; thus, methods at the same step are compared under **matched rollout budgets**. Solid lines show the mean accuracy and shaded regions denote one standard deviation across seeds. Observation: *DGG consistently outperforms the single-use baseline in terms of sample efficiency while maintaining training stability across all random seeds.*

G Analysis of DGG's Dynamic Reuse and Runtime Efficiency

This section provides additional analyses of how DGG behaves in practice, focusing on its dynamically selected reuse depth and the resulting wall-clock efficiency.

G.1 Dynamic Reuse Behavior

Reuse-count distribution. Table 3 reports the empirical distribution of realized reuse counts across all task–model settings. For each rollout batch, DGG performs between one and $K = 4$ policy updates, where reuse either terminates upon gradient gating or reaches the maximum reuse count. The resulting distributions span multiple reuse depths in every setting, with average reuse ranging from 2.43 to 2.92. Moreover, the distributions vary substantially across tasks and models, demonstrating that DGG does not reduce to a fixed reuse schedule.

Table 3: **Distribution of DGG's realized reuse counts** across four tasks and two models. Each entry reports the proportion of rollout batches receiving the corresponding number of policy updates.

Task	Model	Reuse 1 (%)	Reuse 2 (%)	Reuse 3 (%)	Reuse 4 (%)	Avg. Reuse
Math	Qwen2.5-7B-Inst.	12.1	36.2	20.5	31.2	2.71
	Qwen3-4B-Inst.	28.7	26.3	18.0	27.0	2.43
ALFWorld	Qwen2.5-7B-Inst.	6.0	45.1	9.5	39.5	2.82
	Qwen3-4B-Inst.	11.6	42.8	24.6	21.0	2.55
WebShop	Qwen2.5-7B-Inst.	17.7	31.0	23.0	28.3	2.62
	Qwen3-4B-Inst.	26.3	30.3	5.7	37.7	2.55
Search-QA	Qwen2.5-7B-Inst.	10.9	29.4	16.5	43.2	2.92
	Qwen3-4B-Inst.	21.5	38.3	11.9	28.3	2.47

Temporal variation. To further examine whether the selected reuse depth changes over the course of training, Table 4 reports the average reuse count over consecutive training windows on the Math task. The reuse depth varies throughout training and follows distinct temporal patterns across the two models. In particular, Qwen3-4B-Instruct gradually reduces its average reuse depth from 2.72 in the first 100 steps to 2.12 at steps 300–399 before increasing again, whereas Qwen2.5-7B-Instruct remains comparatively stable around 2.6–2.9. These results further illustrate the stage-dependent behavior of DGG.

Table 4: **Temporal variation of DGG's reuse depth** on the Math task. We report the average reuse count over consecutive training windows.

Model	Steps 0–99	Steps 100–199	Steps 200–299	Steps 300–399	Steps 400–495
Qwen2.5-7B-Inst.	2.59	2.87	2.75	2.71	2.64
Qwen3-4B-Inst.	2.72	2.64	2.28	2.12	2.39

G.2 Absolute Runtime Breakdown

Table 5 reports the absolute wall-clock time required to reach the converged performance of the single-use GRPO baseline, decomposed into rollout generation and policy optimization. Across all settings, DGG substantially reduces rollout-generation cost. Although sample reuse may introduce additional policy-update cost, this increase is consistently outweighed by the rollout savings, resulting in $1.31\times$–$2.14\times$ end-to-end wall-clock speedups. The gain is particularly pronounced on agentic tasks, where multi-turn environment interaction makes rollout generation substantially more expensive.

Table 5: **Absolute wall-clock time breakdown** (hours) for GRPO and GRPO+DGG. Times are measured until reaching the converged performance of the single-use GRPO baseline.

Task	Model	Method	Rollout (h)	Update (h)	Total (h)	Speedup
ALFWorld	Qwen2.5-7B-Inst.	GRPO	5.70	0.48	6.18	1.00×
		GRPO+DGG	**2.63**	0.62	**3.25**	**1.90**×
ALFWorld	Qwen3-4B-Inst.	GRPO	13.18	1.48	14.66	1.00×
		GRPO+DGG	**5.32**	1.53	**6.85**	**2.14**×
WebShop	Qwen2.5-7B-Inst.	GRPO	4.84	1.45	6.29	1.00×
		GRPO+DGG	**1.95**	1.53	**3.48**	**1.81**×
WebShop	Qwen3-4B-Inst.	GRPO	3.73	1.31	5.04	1.00×
		GRPO+DGG	**2.03**	1.81	**3.84**	**1.31**×
Search-QA	Qwen2.5-7B-Inst.	GRPO	12.80	2.56	15.36	1.00×
		GRPO+DGG	**5.91**	3.45	**9.37**	**1.64**×
Search-QA	Qwen3-4B-Inst.	GRPO	19.08	3.27	22.35	1.00×
		GRPO+DGG	**8.63**	3.65	**12.28**	**1.82**×
Math	Qwen2.5-7B-Inst.	GRPO	5.37	7.39	12.76	1.00×
		GRPO+DGG	**2.03**	7.28	**9.31**	**1.37**×
Math	Qwen3-4B-Inst.	GRPO	9.68	9.11	18.79	1.00×
		GRPO+DGG	**3.94**	9.02	**12.96**	**1.45**×

H Discussion: Relationship between DGG and SFPO

H.1 Mechanistic Comparison

SFPO (Wang et al., 2026) is an update-stage sample-reuse method designed to improve the reliability of multi-step policy optimization. It performs multiple fast updates on a reused rollout batch and applies a reposition step to interpolate the updated policy toward the original policy, thereby reducing the accumulated off-policy drift. SFPO further uses an entropy-based criterion to determine whether the slow-fast procedure remains active in subsequent training iterations.

Although SFPO and DGG both improve the efficiency–stability trade-off of rollout reuse, they address reuse control **at different granularities**. SFPO uses a **predefined fixed reuse count** for each rollout batch and focuses on stabilizing the policy updates generated across these repeated updates by correcting accumulated drift. In contrast, DGG does not prescribe how many times each batch should

be reused; it **dynamically determines the realized reuse count for each rollout batch** by detecting harmful updates before the optimizer step.

This difference results in distinct adaptive-control mechanisms. In SFPO, the reuse depth K and interpolation strength α are **globally predefined**. Its entropy criterion only determines whether the slow-fast procedure remains active in subsequent training iterations, but **does not change the reuse count of the current rollout batch**. DGG instead performs **fine-grained per-batch reuse control**: it monitors the `lm_head` gradient during reuse and dynamically determines the realized reuse depth of each batch, discarding anomalous updates and terminating reuse once harmful gradients emerge.

Therefore, SFPO and DGG address **different aspects of safe rollout reuse**: SFPO stabilizes the optimization trajectory under a predefined reuse count, whereas DGG dynamically determines whether reuse of the current batch should continue.

H.2 Empirical Comparison

To empirically evaluate the difference discussed above, we compare DGG against the full SFPO method under matched training configurations. We use the official SFPO implementation with its recommended configuration, including the complete fast–reposition–slow procedure and entropy control. All other RL configurations are kept identical.

Performance comparison. Figure 12 compares SFPO and DGG on search-augmented QA task. Compared with Naive Reuse, SFPO substantially mitigates the severe early collapse and sustains performance improvement for considerably longer, confirming the effectiveness of its reposition-based drift correction. Nevertheless, **SFPO shows late-stage performance degradation across all four tasks**, whereas **DGG maintains stable improvement throughout training**.

This behavior is consistent with the mechanistic distinction discussed above. SFPO performs a **predefined number of reuse updates** and stabilizes the resulting optimization trajectory through repositioning; it does not determine whether an individual batch has already reached the point where further reuse becomes harmful. Consequently, even though repositioning effectively suppresses accumulated policy drift, **harmful updates may still be executed** when the fixed reuse depth exceeds what the current batch can safely support. DGG instead **dynamically determines the sample-reuse count for each rollout batch** based on the observed `lm_head` gradient dynamics, discarding anomalous updates and terminating reuse once further updates become harmful.

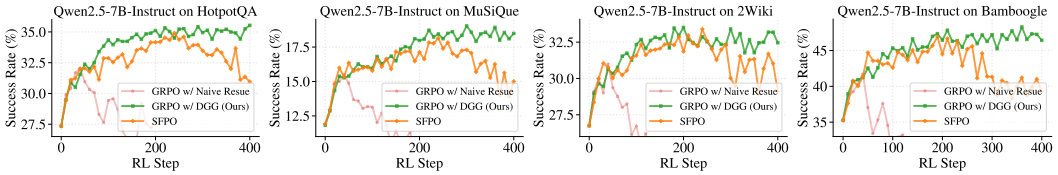

Figure 12: **Performance comparison between DGG and SFPO on search-augmented QA task** using Qwen2.5-7B-Instruct. All methods use the same underlying RL configuration, differing only in their sample-reuse mechanisms. Each RL step corresponds to one fresh rollout batch; thus, methods at the same step are compared under **matched rollout budgets**. Observation: *SFPO substantially mitigates the instability of naive reuse, but still exhibits late-stage performance degradation; DGG dynamically determines the reuse count for each rollout batch and maintains stable improvement throughout training.*

Sensitivity to the maximum reuse count. Beyond the direct performance comparison, we further compare the sensitivity of SFPO and DGG to the maximum reuse count K. Figure 5(c) in the SFPO paper shows that increasing K from 3 to 5 leads to markedly different training trajectories, indicating notable sensitivity to the reuse count. In contrast, Figure 5 shows that DGG remains stable across $K \in \{2, 4, 6, 8\}$, with particularly small performance differences among $K \in \{4, 6, 8\}$.

This contrast is consistent with the different reuse-control mechanisms. SFPO applies a **globally predefined reuse count**, making its training behavior directly dependent on the choice of K. DGG instead **dynamically determines the realized reuse count for each rollout batch**, with K serving

only as the maximum allowed reuse depth. This per-batch adaptation makes DGG less sensitive to the maximum-reuse setting.

I More Experimental Details

I.1 Training Details.

Mathematical reasoning experiments are conducted on 32 NVIDIA H800 80GB GPUs, and agentic experiments (ALFWorld, WebShop, and search-augmented QA) are conducted on 16 NVIDIA H800 80GB GPUs. All methods are configured with identical hyperparameters within each task. The learning rate is $1e{-6}$ and the sampling temperature is 0.7 throughout. For mathematical reasoning, the maximum prompt length is 2048 tokens, the maximum response length is 4096 tokens, the group size is 16, the prompt batch size is 512, and the KL coefficient is 0. For ALFWorld, the maximum prompt length is 8192 tokens, the maximum response length is 8192 tokens, the group size is 8, the KL coefficient is 0.001, the prompt batch size is 16, and the maximum interaction steps is 50. For WebShop, the maximum prompt length is 4096 tokens, the maximum response length is 4096 tokens, the group size is 8, the KL coefficient is 0.001, the prompt batch size is 32, and the maximum interaction steps is 30. For search-augmented QA, we use E5 (Wang et al., 2022) as the retriever, following (Feng et al., 2026). The maximum prompt length is 2048 tokens, the maximum response length is 8192 tokens, the group size is 8, the KL coefficient is 0.001, the prompt batch size is 64, and the maximum interaction steps is 10.

I.2 Implementation Details of DGG.

At each gradient step, we compute the `lm_head` gradient energy $g_t = \|G_t^{\text{lm}}\|_F^2$ and its step-wise increment $\Delta g_t = g_t - g_{\text{ref}}$, where g_{ref} denotes the gradient energy of the most recent accepted update. The running statistics (μ_t, σ_t) are maintained as the sample mean and standard deviation of Δg_t over a sliding window of the most recent $W = 20$ past increments (excluding the current step), and the Z-score $z_t = (\Delta g_t - \mu_t)/(\sigma_t + \varepsilon)$ with $\varepsilon = 10^{-8}$ is evaluated against the threshold τ.

Gating is activated only when (i) the window is fully populated ($t \geq W$) and (ii) the current reuse index satisfies $k > 1$; under these conditions, $z_t > \tau$ triggers gradient discard and reuse termination as described in Section 5. An intercepted update is not treated as a valid historical state: its gradient energy is neither added to the running statistics nor used to update g_{ref}. All other steps follow standard GRPO. The full procedure is summarized in Algorithm 1. We set the maximum reuse $K = 4$ and select the anomaly threshold τ from the candidate set $\{0.1, 0.5, 1.0\}$.

J Relative Weight Change throughout Full RL Training

The main paper reports the relative weight change during the first 200 RL steps to clearly visualize the onset of the DWD phenomenon under sample reuse. To rule out the possibility that the abrupt `lm_head` surge is caused by the increased number of gradient updates rather than sample reuse itself, we extend the profiling on Math500 to the full training trajectory under the GRPO w/o Reuse regime. As shown in Figure 13, the relative weight change of all components—including the `lm_head`—grows smoothly and remains stable throughout the entire training process, with no abrupt surge. This serves as a controlled comparison to the sample-reuse setting (Figure 1): under matched gradient-update budgets but without sample reuse, the `lm_head` surge does not occur, confirming that sample reuse—not the number of updates—is the direct cause of the abrupt weight divergence underlying DWD.

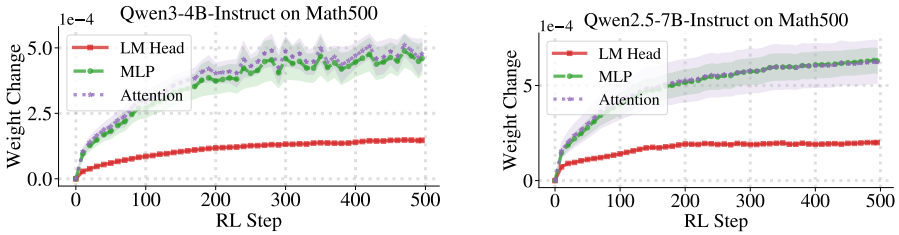

Figure 13: **Relative weight change throughout the full RL training process** on Qwen3-4B-Instruct and Qwen2.5-7B-Instruct (Math500), under the GRPO w/o Reuse regime. Observation: *Without sample reuse, the relative weight change of all components—including the* `lm_head`*—grows smoothly throughout training, confirming that the abrupt* `lm_head` *surge observed under sample reuse is caused by reuse itself rather than by the increased number of gradient updates.*