Selective Rollout: Mid-Trajectory Termination for Multi-Sample Agent RL

Zhiyuan Zhai
Fudan University
22110720067@m.fudan.edu.cn

Xin Wang
Fudan University
xwang11@fudan.edu.cn

Abstract

Group Relative Policy Optimization (GRPO) samples a small group of parallel rollouts for every training prompt and uses their within-group reward spread to compute per-trajectory advantages. In agentic environments each rollout is a long multi-turn dialogue with one LLM call per step, so this multi-sample multiplier dominates the total training cost. When every rollout of a prompt ends with the same reward, the group has zero reward variance and contributes no group-relative reward gradient, so the extra rollouts add no information; such groups are common in practice (typically around 40% of all groups), so the wasted-compute fraction is substantial rather than marginal. Existing methods filter such groups at the prompt level, either after their rollouts are paid for or before any rollout begins, but both decide without using information that becomes available during the rollout itself. We instead ask whether the in-group divergence between the partial trajectories at an intermediate step can already predict that the group will be zero-variance: when the parallel rollouts have already converged on the same action prefix, the group is on track to produce a single reward, and we can stop early. We propose a one-parameter gate that stops a group when the mean pairwise prefix edit distance between its partial action sequences falls below a threshold. On a 60-iteration on-policy GRPO run on ALFWorld with Qwen2.5-7B, averaged over four random seeds, the gated arm finishes $\mathbf{10.7}\%$ faster in wall-clock (bootstrap 95% CI excludes 0) and shifts held-out success rate on 50 unseen tasks by $+\mathbf{2.5}$ percentage points (not statistically significant across four seeds). The gate is predictive rather than lossless in general; false cuts and task-specific calibration remain important limitations.

1 Introduction

Group-Relative Policy Optimization (GRPO [1, 2]) has become the dominant family of RL methods for fine-tuning language models on preference and outcome rewards [3], replacing the learned value baseline of PPO [4] with the empirical mean reward of a small group of parallel rollouts of the same prompt. The per-prompt cost of a GRPO iteration grows linearly with the number of rollouts in the group. In agentic environments such as ALFWorld [5], WebShop [6], or AgentBench [7] this multiplier matters more than usual: each rollout is a multi-turn dialogue, often 20–30 environment steps long with one LLM call per step, so a single rollout already takes minutes of wall-clock. Multiplying that by eight

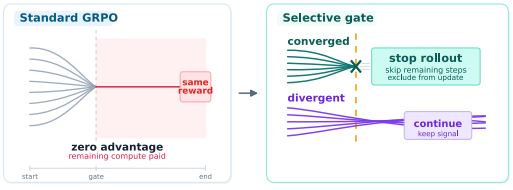

Figure 1: Selective rollout. Standard GRPO pays the full cost of converged groups (*left*); our gate predicts low-variance groups and stops them mid-rollout (*right*); false cuts remain possible.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

or sixteen rollouts per prompt makes the rollout phase the dominant cost in the entire training pipeline ($\sim$95% of wall-clock in our setup). A *rollout* is one complete interaction; a *group* contains independent samples for the same task. For example, an agent observes "mug 1 on countertop 1", emits `take mug 1 from countertop 1`, and receives "you are holding mug 1". One such command and response form an environment step; the gate compares command prefixes across rollouts.

Not all of this compute is paying for new information. When every trajectory in a group ends with the same reward, the group has zero reward variance: the within-group normalisation sets every trajectory's advantage to exactly zero, so the group contributes nothing to the policy gradient. Such groups are common in practice: with Qwen2.5-7B on ALFWorld, 39% of the offline groups we collect are zero-variance (25 all-fail and 14 all-success out of 100; binary terminal rewards), and the rate hovers around 40% averaged across the 60 iterations of the on-policy training run in §4.3. Existing approaches recover this cost at the prompt level: DAPO [8] filters zero-variance groups out of the gradient batch *after* the rollouts have been paid for, while GRESO [9] predicts uninformative prompts *before* any rollout begins, using cross-epoch reward consistency on mathematical reasoning benchmarks. Both decide at the prompt level and leave the within-rollout compute untouched. The 39% rate is specific to this buffer: shared task difficulty under sparse binary rewards makes both uniform failure and uniform success common.

This paper asks: can we predict, partway through a group's rollout, that the group is going to be zero-variance? If yes, we can stop the rollout early and recover the rest. In an agent rollout, trajectories interact with the environment over many steps, exposing partial actions and observations. Long single-turn reasoning also permits token checkpoints, but requires different signals; a full autoregressive rollout is not a single forward pass. Our experiments focus on multi-turn agent rollouts. Our hypothesis is that when these partial trajectories have already converged on a single action sequence at the chosen intermediate step, the group is on track to be zero-variance: the policy has effectively committed to one behaviour, and the remaining steps will only confirm that commitment. Conversely, divergent prefixes signal a group that is still meaningfully exploring and worth letting run.

We test this hypothesis on 100 ALFWorld groups with group size eight and find that a single threshold on the mean pairwise prefix-edit distance suffices. The same gate then drops into a 60-iteration on-policy GRPO training loop, where it produces a measurable wall-clock saving and a small, statistically inconclusive held-out improvement. We also give a quantitative account of the loss-normalization effect of dropping groups. This effect is shared with post-rollout filtering and does not establish a causal explanation for the held-out difference.

Contributions.

- **A one-parameter mid-rollout gate** based on in-group prefix-edit divergence at step K. On $N = 100$ ALFWorld groups with $G = 8$, the gate recovers $\mathbf{14.0}\%$ of fixed-horizon rollout-step budget at offline precision 0.81 ($\mathbf{11.3}\%$ from true positives) while preserving $\mathbf{96.7}\%$ of the GRPO advantage L^2-norm.
- **End-to-end on-policy validation.** Across $n = 4$ seeds of a 60-iteration on-policy GRPO training run, the gated arm finishes $\mathbf{10.7}\%$ faster (bootstrap 95% CI excludes 0) and shifts held-out success on 50 unseen tasks by $+\mathbf{2.5}$ pp.
- **Analysis of loss normalization and false cuts.** We separate suffix-generation savings from changes to batch normalization and report the retained advantage norm and observed gradient magnitude without attributing the held-out difference to them.

We defer the full related-work discussion to App. A.

2 Preliminaries

2.1 GRPO

Let π_θ be a language-model policy and $\mathcal{D}$ a distribution over agent prompts. For prompt $x \sim \mathcal{D}$, GRPO [1] samples a group of G rollouts $\{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_G\}$ from π_θ, each a sequence of (observation, action, reward) triples up to horizon $T_{\max}$. Let $r_i = R(\boldsymbol{\tau}_i)$ be the terminal reward. The group-relative

advantage is

$$A_i = \frac{r_i - \bar{r}}{\sigma_r + \varepsilon}, \quad \bar{r} = \frac{1}{G}\sum_j r_j, \quad \sigma_r = \sqrt{\frac{1}{G}\sum_j (r_j - \bar{r})^2}. \tag{1}$$

Intuitively, A_i is the within-group z-score of r_i: it is positive when trajectory i scored above the group mean and the policy should be pushed *toward* reproducing it, negative when below the mean (push away), and zero when $r_i = \bar{r}$. The conceptual trajectory-level REINFORCE-style [10] policy-gradient loss is $\mathcal{L}(\theta) = -\frac{1}{N}\sum_i A_i \log p_\theta(\boldsymbol{\tau}_i)$, averaged over all N trajectories in the gradient batch (agent-specific teacher-forcing in App. C.2).

A group is *zero-variance* when σ_r is zero, that is, when all G trajectories in the group end with the same reward. For such a group the advantage A_i is zero for every i, so the group contributes nothing to the group-relative reward gradient. This statement excludes separate regularizers such as a KL penalty.

This zero contribution causes a second, more subtle effect that will be important later. The policy-gradient loss above is the *mean* over all N trajectories in the batch, so it divides by N. A trajectory whose advantage is zero does not add to the numerator of this mean, but it still occupies one of the N slots in the denominator. The result is that the gradient contributed by the trajectories with non-zero advantage is shrunk by a factor equal to the fraction of non-zero-advantage trajectories in the batch. If half the trajectories in a batch have advantage zero, the effective gradient is half what it would be on a batch of only non-zero-advantage trajectories. We return to this effect in §4.4.

2.2 Compute decomposition in agent RL

We distinguish **wall-clock** (elapsed execution time), **environment actions** (executed commands), and generated **tokens**. The original fixed-horizon proxy counts potential environment steps against $NGT_{\max}$, assuming equal step cost; it is not a token count or a measured wall-clock saving. Actual-length action savings count only remaining actions in completed trajectories. Each GRPO iteration in our setting consists of three phases:

- **Rollout (sampling)**. The policy generates the G trajectories in parallel, with autoregressive token generation within each trajectory. At K, termination is one all-or-none group decision; we do not prune individual trajectories using unknown final advantages. For ALFWorld with $T_{\max} = 30$ and $G = 8$ this is on the order of 240 generation calls per prompt, with no gradient computation. In our on-policy training runs this phase consumes $\sim 95\%$ of total wall-clock.
- **Advantage computation**: Eq. 1. Negligible cost.
- **Training (gradient update)**. For sampled environment-step items, the trainer evaluates the prefix plus current assistant response, averages response-token log-probabilities, and backpropagates the advantage-weighted surrogate to update the LoRA parameters (App. C.2). This phase incurs forward/backward costs and activation memory only for trajectories retained in the training batch; its relative cost depends on sequence length, batching, and hardware.

A zero-variance group wastes all three phases: it pays the full rollout cost, computes a zero advantage, and (unless filtered) contributes a zero loss with the full gradient-update cost. The gate we develop next eliminates the post-K tail of the rollout cost in addition to the training cost.

3 Selective-rollout gate

3.1 Setup

ALFWorld [5] is a text environment for household instruction-following: the agent reads a scene description and issues a discrete command. We use $N = 100$ groups (800 trajectories) from `valid_seen`. The six task families are placing one object, placing two objects, cleaning, heating, cooling, and examining an object under a light, with $24, 20, 18, 11, 19, 8$ groups, respectively. The policy is Qwen2.5-7B-Instruct [11], sampled at temperature $T = 0.7$ with group size $G = 8$ and rollout horizon $T_{\max} = 30$. Across the 100 groups we observe 25 all-fail, 61 mixed, and 14 all-succeed groups (a 39% zero-variance rate). All released code and result tables are available at `https://github.com/zhiyuanZhai20/selective-rollout`; the public snapshot does not include all raw rollouts or the additional validation artifacts.

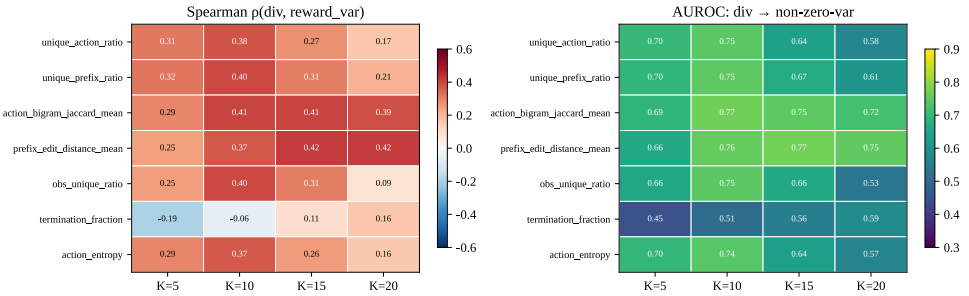

Figure 2: Spearman ρ (left) and AUROC for non-zero-variance classification (right) for each (metric, K) cell, $N=100$, $G=8$. The single-axis signal peaks at $K \in \{10, 15\}$ and degrades at the extremes.

3.2 Mid-rollout signals and the gate

Let $a_{i,t}$ be the parsed textual environment command of trajectory i at step t, excluding observations and `Thought:` text. At an intermediate step $K \leq T_{\max}$ we want a scalar measure of how much the G partial trajectories disagree with each other so far. The signal we use throughout the paper is the **mean pairwise prefix edit distance** $d_K \in [0, 1]$: for each pair (i, j) we take the Levenshtein [12] edit distance between the action sequences $a_{i,1:K}$ and $a_{j,1:K}$, normalise by the longer of the two lengths, and average over all $\binom{G}{2}$ pairs. Prefixes ending before K are padded with an end symbol. $d_K = 0$ means identical padded prefixes; $d_K = 1$ means maximal normalized edit distance, not necessarily disjoint action vocabularies. We considered six other in-group divergence signals (action-bigram Jaccard, unique-prefix ratio, unique-action ratio, action entropy, observation-unique ratio, and termination fraction) and pick d_K because it is among the strongest predictive signals at $K \in \{10, 15\}$ (§3.3, Fig. 2) and is the most directly interpretable. Full definitions and the side-by-side comparison are in App. B. To label groups in the predictive analysis below we use the binary label $\mathbf{1}[\sigma_r = 0]$ and the three-way label $\ell \in \{\text{all_fail}, \text{mixed}, \text{all_succeed}\}$, both computed only after the rollout finishes.

The gate. We propose a one-parameter gate that is evaluated at step K and decides whether to terminate the group:

$$\textsc{Cut}_{K,d_L}(\boldsymbol{\tau}_{1:G}) = \Big[\, d_K < d_L \,\Big]. \tag{2}$$

The gate fires when the G partial trajectories have sufficiently similar action prefixes at step K. This predicts, rather than proves, that the completed group has zero reward variance. This happens in two situations: either the policy is confident about the task and all G rollouts are heading toward the same successful behaviour, or the policy has gotten stuck in an early loop and all G rollouts are repeating the same failing actions. When the gate fires, the group is removed from the GRPO update entirely: any trajectories in the group that are still running are stopped at step K, and the entire group is dropped from the gradient batch (i.e. none of its trajectories enter $\mathcal{L}$ defined in §2.1). The compute saving is therefore twofold: the post-K rollout cost (up to $T_{\max} - K$ environment steps per active trajectory) and the training-step cost (forward and backward pass over those trajectories). Implementation details for the rollout-loop integration are in App. C.1. The precision floor controls empirical false-cut frequency, not gradient bias or norm. The deterministic gate used in the ALFWorld experiments can discard useful gradients; App. D.2 separates this heuristic from an optional randomized-continuation correction.

3.3 Mid-rollout divergence correlates with reward variance

For each of the seven divergence measures defined in §3.2 and each evaluation step $K \in \{5, 10, 15, 20\}$, we ask two questions about the 100 groups in our buffer. First, how strongly does the divergence value at step K correlate (in rank order) with the final reward variance of the group? We measure this with Spearman's rank correlation ρ. Second, how well does the same divergence value distinguish zero-variance groups from non-zero-variance groups when treated as a binary classifier score? We measure the area under the receiver operating characteristic curve (AUROC): 0.5 is chance ranking and 1 is perfect ranking. Here the positive class is non-zero variance, so 0.77 means a random mixed group outranks a random zero-variance group about 77% of the time; it does not mean 77%

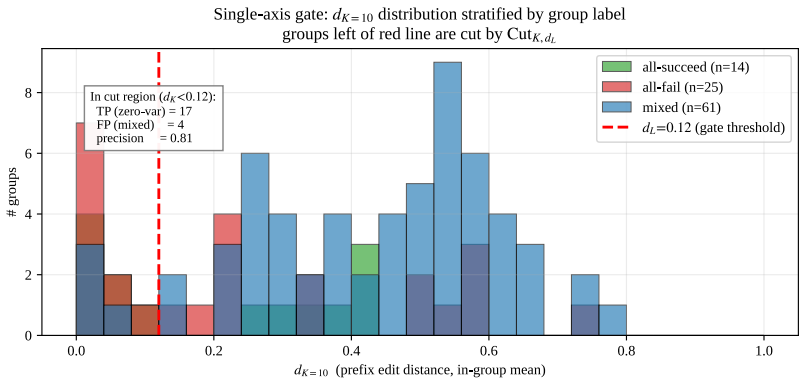

Figure 3: Distribution of $d_{K=10}$ on the $N = 100$ buffer, coloured by the group's final outcome (determined only after the full rollout completes, by inspecting the multiset of G final rewards). At $d_L = 0.12$ (red dashed line) the gate catches $7/14$ all-succeed and $10/25$ all-fail groups (the 17 true positives) plus $4/61$ mixed groups (false positives). The high-d_K tail of the all-fail cluster overlaps with the mixed cluster, so raising d_L would break the precision floor (§5).

classification accuracy. The full 7×4 heat map is shown in Fig. 2, with the underlying table in App. E.3.

The signal is strongest at $K \in \{10, 15\}$. For prefix edit distance (d_K) at $K = 15$, ρ reaches 0.42 ($p = 1.4 \times 10^{-5}$) and AUROC reaches 0.77. Most sequence-diversity measures have comparable ranking quality around $K = 10$–20; termination fraction is less predictive. Later checkpoints may retain predictive value but leave fewer actions to save. We choose d_K for the gate because it is the most consistent of the seven across K values and because it has a direct interpretation as an average pairwise edit distance. Breaking the same d_{10} signal down by ALFWorld task type shows that the signal generalises across tasks but unevenly. The median per-type AUROC is 0.76, with values ranging from 0.48 to 1.00 across the six task types. If we are allowed to tune the metric and the evaluation step K separately for each task type, the worst case rises to AUROC 0.74 and the median to 0.89 (App. E.5). These are exploratory in-buffer comparisons on only 8–24 groups per family, not held-out per-family guarantees; the perfect AUROCs involve only one or two zero-variance groups.

3.4 Where the signal lives: $d_{K=10}$ stratified by group label

Figure 3 explains how the cut region splits across the three group labels. Both all-succeed and all-fail groups have a sub-population at low d_K, and the gate at $d_L = 0.12$ catches both: of the 17 true-positive cuts, 7 are from the 14 all-succeed groups and 10 are from the 25 all-fail groups. The all-succeed distribution is more tightly concentrated at low d_K and the gate catches half of it ($7/14$); the all-fail distribution has a heavier right tail that overlaps with the mixed cluster, so the gate only reaches its low-d_K end ($10/25$). The mixed cluster (the 61 groups with both wins and losses) has a right tail at higher d_K that the gate correctly leaves alone, and a left tail at lower d_K that produces the four false positives we discuss in the ablation below. Raising d_L to reach the remaining $\sim$15 high-d_K all-fail groups would pull in many more mixed groups as false positives and break the precision floor.

3.5 Choosing K and d_L, and the savings–signal tradeoff

Table 1 sweeps d_L at $K = 10$. The selected $d_L = 0.12$ cuts 21 groups (17 TPs, 4 FPs), giving 14.0% gross and 11.3% true-positive fixed-horizon proxies. Accounting instead for actual completed trajectories gives $2{,}170/18{,}197 = 11.93\%$ gross and $1{,}723/18{,}197 = 9.47\%$ true-positive action savings. The distinction matters because many trajectories finish before $T_{\max}$. False cuts affect $4/61 = 6.56\%$ of mixed groups. K and d_L were selected on this buffer, so these are calibration results, not out-of-sample risk guarantees.

Savings vs gradient signal preserved. TP cuts remove zero group-relative reward advantages. The retained advantage-vector ratio $\|A_{\text{kept}}\|_2 / \|A_{\text{full}}\|_2$ is 96.7% at $(K, d_L) = (10, 0.12)$. This is an

Table 1: Single-axis gate $\textsc{Cut}_{K,d_L}$ at $K = 10$ on $N = 100$ groups (39 zero-variance) for varying threshold d_L. Columns: **cut** = total groups the gate fires on (= TP+FP), **TP** = correctly cut zero-variance groups, **FP** = mistakenly cut non-zero-variance groups, **prec.** = TP/cut, **recall** = TP/39, **safe** = the fixed-horizon step-budget proxy from TP cuts alone, $\text{TP}\cdot(T_{\max}-K)/(N\cdot T_{\max})$ (TPs have advantage exactly 0, so dropping them removes no gradient signal), **raw** = the fixed-horizon step-budget proxy from all cuts, $\text{cut}\cdot(T_{\max}-K)/(N\cdot T_{\max})$ (this proxy assumes equal step cost and full-length continuations; it is not a measured token or wall-clock saving). Operating points are subject to a precision floor of ≥ 0.80 as an empirical calibration criterion (discussion in App. C.5).

d_L	cut	TP	FP	prec.	recall	safe (%)	raw (%)	note
0.05	14	11	3	0.79	0.28	7.3	9.3	below floor
0.08	19	15	4	0.79	0.38	10.0	12.7	below floor
0.10	20	16	4	0.80	0.41	10.7	13.3	at floor
0.12	**21**	**17**	4	**0.81**	0.44	**11.3**	**14.0**	**chosen for Tier 3**
0.14	24	18	6	0.75	0.46	12.0	16.0	below floor
0.18	25	19	6	0.76	0.49	12.7	16.7	below floor

Table 2: Experimental setup across the three integration tiers. In Tier 1 the same 100 tasks are run twice (baseline and gated) with matched random seeds. In Tier 3 the trained policy weights are loaded back into the inference engine after every iteration.

	Tier 1: Rollout	**Tier 2: Off-policy**	**Tier 3: On-policy**
purpose	rollout-time saving	training-time saving	full end-to-end loop
prompt budget	100 tasks ($\times$2 arms)	100 tasks (replay buffer)	10 prompts/iter $\times$ 60 iters
gradient updates	none	20 LoRA steps	60 iterations
trajectory source	current θ	θ_0 (frozen)	current θ_t
held-out eval	—	—	50 tasks, every 10 iters

Common to all tiers

Base model: Qwen2.5-7B-Instruct [11]. Environment: ALFWorld [5] valid_seen, six task types, $T_{\max} = 30$, sample $T = 0.7$.
Group size $G = 8$ unless stated. Gate $(K, d_L) = (10, 0.12)$; Tier 2 also uses $\tau_K \geq 0.90$.
Hardware: NVIDIA RTX 6000 Ada (48 GB).
Per-tier LoRA, optimiser, and random seed(s) are listed in App. C.4, Table 6.

advantage-norm diagnostic, not a bound on the gradient vector: score-function magnitudes, directions, normalization, and regularization also matter.

4 Online integration: three-tier validation

We validate the gate at three progressively more realistic integration points (Table 2): **Tier 1** measures the rollout-phase wall-clock saving alone, with no policy updates, an A/B measurement of rollout time under the recorded runs. **Tier 2** adds 20 GRPO training steps over a fixed rollout buffer to isolate the training-phase saving (the policy does not change between rollouts). **Tier 3** adds the full closed loop (60 on-policy GRPO iterations in which the policy is updated after each group of rollouts and the next iteration's rollouts use the updated policy) and adds held-out evaluation against 50 unseen tasks.

The Tier-3 held-out split of 50 `valid_seen` tasks is disjoint by task instance from the 10 prompts used per training iteration.

4.1 Tier 1 — Rollout-only A/B

We compare two ways of generating rollouts on the same 100 ALFWorld prompts, in a fresh A/B re-run rather than a re-analysis of the §3.1 offline buffer. The **baseline** run rolls each task out for up to 30 environment steps or natural termination, sampling $G = 8$ trajectories per task. The **gated** run does the same, but at step 10 it pauses each group, computes the divergence d_{10} between its eight partial trajectories, and stops the group if $d_{10} < 0.12$. Up to 20 remaining steps are skipped. The archived Tier-1 implementation uses the single-axis rule (the termination-fraction backup is disabled). Although nominal seeds match, the archived action prefixes do not match for most interrupted groups; sampling nondeterminism and execution order preclude using the baseline as their exact missing suffix.

The run interrupts $20/100$ groups and reduces summed task durations by 13.25%, from $2{,}678.6$ to $2{,}323.6$ s. A paired task bootstrap with $10{,}000$ resamples (seed 42) gives a 95% interval of $[7.47, 19.03]\%$. This timing result does not establish the terminal precision of the interrupted groups; the 0.81 precision above comes from the separate complete offline buffer.

Table 3: **Tier 1.** Summed per-task rollout time on the same 100 ALFWorld tasks with $G = 8$ and a checkpoint at $K = 10$. The single-axis gate is used; these sums are not parallel supervisor elapsed time. Task-bootstrap 95% CI for saving: $[7.47, 19.03]\%$. Unmatched sampled prefixes prevent a counterfactual precision estimate from these two runs.

	baseline	gated	Δ
sum of task times (s)	2,678.6	2,323.6	$\mathbf{-13.25}\%$
mean per-task (s)	26.8	23.2	-13.25%
groups cut (of 100)	0	20	$+20$

4.2 Tier 2 — Off-policy GRPO training A/B

Tier 1 measured only the time spent generating rollouts. We next check that the gate also reduces the cost of the training step itself. We re-use the same 100 ALFWorld groups (each is a prompt together with its $G = 8$ rolled-out trajectories and their rewards) as a fixed buffer, and run a 20-step GRPO training loop over this buffer. At each of the 20 training steps we sample 4 groups from the buffer, compute the group-relative advantages from Eq. 1, and take one AdamW [14] update on the LoRA parameters of Qwen2.5-7B.

In the gated arm we apply the OR-rule extension before forming the training batch. The gate condition uses the d_{10} value already recorded in the buffer for each group; a group whose $d_{10} < 0.12$ or $\tau_{10} \geq 0.90$ is dropped from this training step's batch, and the loss is averaged over selected environment-step items. Because the buffer is fixed and the policy is not used to generate new rollouts between training steps, the only difference between the two arms is whether or not the cut groups enter each step's gradient batch. This isolates the training-side effect of the gate from any rollout-generation differences.

Across the 20 training steps and 4 groups per step, the gate drops 18 of the 80 groups it sees, which matches the prediction from the offline analysis. The mean per-step gradient L^2-norm *rises* by 14% in the gated arm (Table 4). This is consistent with a normalization effect, but the deterministic gate can also drop mixed groups, so an unchanged gradient numerator is not guaranteed. The training-phase wall-clock falls by 32.2%; this is a fixed-buffer diagnostic, not an end-to-end on-policy saving. The per-step trace is in App. E.7.

4.3 Tier 3 — On-policy GRPO training A/B with held-out eval

Tier 3 is the full closed-loop on-policy GRPO run: at each of 60 iterations we draw 10 ALFWorld prompts, sample $G = 8$ trajectories per prompt (the gate is active in the gated arm and inactive in the baseline arm), compute group-relative advantages (Eq. 1), and take one AdamW update on the LoRA parameters of Qwen2.5-7B; updated weights are hot-swapped into the inference engine before the next iteration (App. C.3). Every 10 iterations both arms are evaluated greedily on 50 unseen `valid_seen` tasks. We repeat the experiment with 4 random seeds, with each baseline run paired to a gated run sharing the same seed so the only difference is the gate (per-iter trace in App. E.8).

The wall-clock saving (Fig. 4 top-right) is consistent across seeds: every seed lies in $[8.95, 14.16]\%$ and the mean is $\mathbf{10.7}\%$ with a bootstrap 95% CI that excludes zero. The gradient L^2-norm rises by $1.16\times$ in the gated arm, consistent with the normalization discussion of §4.4. Held-out success (Fig. 4 top-left): both arms start from the same pre-trained checkpoint at iter 0, so the two iter-0 points coincide per-seed (their across-seed std of 3.0 pp reflects only the seeded 50-task held-out subset); the curves diverge with training and reach 41.5% (baseline) vs. $\mathbf{44.0}\%$ (gated) at iter 60, a $+2.5$ pp gap (per-seed $+4, +6, -2, +2$ pp). The shift is directionally favourable but not statistically significant at $n = 4$. The bottom-right panel of Fig. 4 compares gate firings and baseline zero-variance counts; without matched full continuations it cannot measure false cuts. The training-reward panel averages only uncut groups, whose outcomes are observed. That conditional average is not an

Table 4: **Tier 2.** Off-policy GRPO training A/B on Qwen2.5-7B + LoRA, 20 steps, 4 groups/step, $G=8$ trajectories per group from a frozen $N=100$ buffer. This diagnostic uses the OR-rule backup $d_{10} < 0.12 \vee \tau_{10} \geq 0.90$.

	baseline	gated	Δ
total step wall-clock (s)	496.0	336.3	$-\mathbf{32.2}\%$
total train items	2,560	1,984	-22.5%
groups cut (out of 80)	0	18	cut rate 22.5%
gradient L^2-norm mean	0.086	0.098	$+14.0\%$

Table 5: **Tier 3.** On-policy GRPO training A/B with the gate ($K=10$, $d_L=0.12$) on Qwen2.5-7B + LoRA, 60 iterations, 10 prompts/iter, $G=8$, averaged over $n=4$ seeds ($\{7, 13, 23, 42\}$). Held-out eval is greedy on 50 `valid_seen` ALFWorld tasks, never seen during training. Per-seed pre-training eval values are $38, 44, 38, 38\%$ (mean $39.5\pm3.0\%$); per-seed held-out delta at iter 60 is $+4, +6, -2, +2$ pp. Standard deviations across the 4 seeds: total wall-clock ±187 / ±534 s; gradient L^2 ±0.001 / ±0.008; held-out eval ±4.4 / ±7.1 pp.

	baseline	gated	Δ
total wall-clock (s)	17,393±187	15,527±534	$-\mathbf{10.7}\pm\mathbf{2.4}\%$
groups cut by gate (of 600)	0	106±18	+106
gradient L^2-norm	0.125±0.001	0.145±0.008	1.16±0.07×
held-out eval, iter 0 (pre-training)	39.5±3.0%	39.5±3.0%	0
held-out eval, iter 60	41.5±4.4%	44.0±7.1%	$+\mathbf{2.5}\pm\mathbf{3.4}$ pp

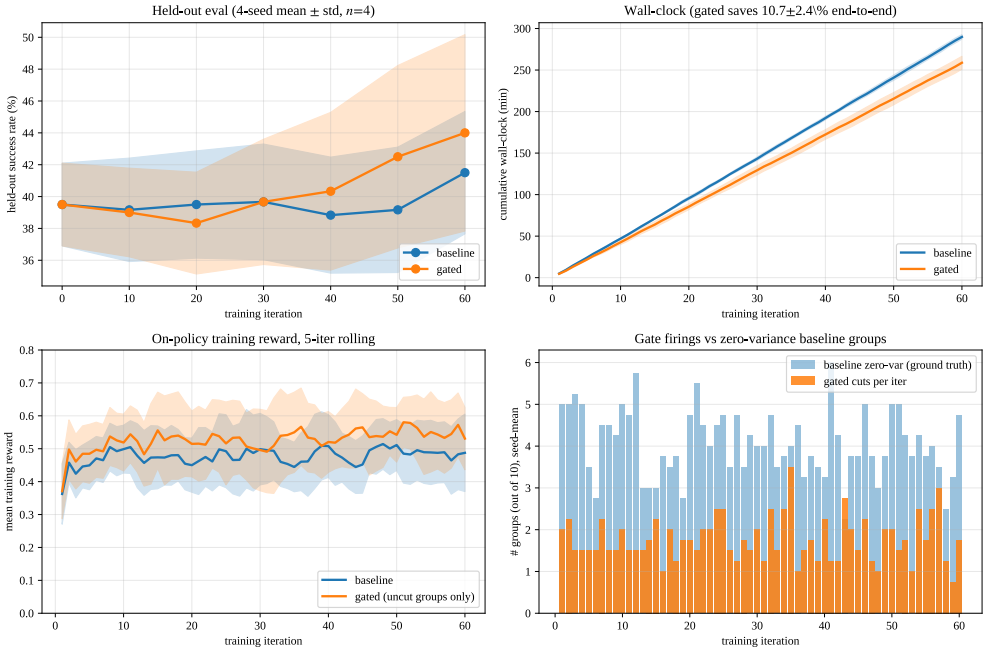

Figure 4: On-policy GRPO A/B on Qwen2.5-7B + LoRA, gate $\textsc{Cut}_{K=10,d_L=0.12}$, averaged over $n=4$ seeds $\{7, 13, 23, 42\}$; bands are ±1 std across seeds. *Top-left:* held-out success on 50 unseen tasks (every 10 iter). *Top-right:* cumulative wall-clock; gated runs 10.7% shorter at iter 60. *Bottom-left:* per-iter mean training reward (5-iter rolling), gated averaged over uncut groups only. *Bottom-right:* per-iter gate cut count (orange) vs. baseline zero-variance count (blue); the gate cut count is compared with the baseline zero-variance count. These aggregate counts do not establish groupwise precision or recall.

unbiased estimate of the full gated batch's terminal reward. Held-out, gate-off evaluation is the relevant policy-quality comparison.

Taken together, the runs show consistent compute savings and a positive but statistically inconclusive held-out difference. They do not establish non-inferiority or a guaranteed policy improvement.

4.4 Loss normalization and gradient magnitude

For a fixed batch, removing only zero-advantage trajectories leaves the reward-gradient numerator unchanged. If the loss is then averaged over the remaining trajectories, its denominator shrinks and gradient magnitude increases. If the original denominator is retained, this rescaling disappears. Post-rollout filtering such as DAPO can produce the same effect; our additional contribution is avoiding suffix generation.

The measured gradient-norm ratio is 1.16 ± 0.07 across four seeds (Table 5). It is compatible with a normalization effect but is not a signal-to-noise measurement or proof of the cause of the $+2.5$ pp held-out difference. False cuts can change the numerator, and AdamW's adaptive state means a gradient-norm ratio is not a parameter-update ratio. Further discussion and a conditional unbiased-continuation identity appear in App. D.1–D.2.

Appendix checks show that the divergence trend persists at $G = 16$ (peak $\rho = 0.37$ versus 0.42 at $G = 8$), while retuning yields a smaller 7.0% fixed-horizon saving (App. E.6). For $G = 8$, the selected gate retains 96.7% of advantage L^2; a uniformly random cut of 21 groups retains 88.9% in expectation (App. E.11).

5 Discussion

Relation to prompt-level filters. DAPO [8] drops zero-variance groups from the gradient batch *after* their rollouts have been generated, so it saves the training-step cost but not the rollout cost. GRESO [9] predicts zero-variance prompts *before* any rollout begins, using cross-epoch reward consistency on math benchmarks; it has no way to use information that becomes available *during* a rollout. Our gate is orthogonal to both: it operates mid-rollout, before the group's reward variance is known, using only the partial trajectories observed up to step K. The two filters compose naturally: our gate cuts the low-d_K end of both convergent-success and convergent-failure groups mid-rollout to save the post-K rollout cost, while DAPO cleans up the high-d_K tail of convergent-failure groups (which our prefix-divergence signal cannot reach without breaking the precision floor; §3.5) by dropping any surviving zero-variance groups from the gradient batch. The rollout accounting for this combination is given in App. E.11, Table 12; end-to-end combined training remains to be evaluated.

Practical use. The main ALFWorld gate fixes $K = 10$ and calibrates one threshold d_L. Computing pairwise edit distances adds $O(G^2K^2)$ work on short command sequences. End-to-end overhead also includes worker coordination; production cancellation and straggler effects require separate validation. Signals and thresholds require calibration when the environment, policy, or prompt distribution changes.

Limitations. The gate misses many high-divergence zero-variance groups and can prematurely stop mixed groups, biasing a deterministic update. The 0.80 floor is an empirical calibration criterion, not a population guarantee. Four ALFWorld seeds support a compute-saving result but not a statistically significant policy improvement. Model and environment are fixed (Qwen2.5-7B + ALFWorld), so transfer beyond this setting is unverified here. The checkpoint K is fixed; longer training and changing trajectory lengths may alter both gate quality and realized savings.

Conclusion. Mid-trajectory group-level gating reduces measured ALFWorld rollout and training costs, with no statistically resolved held-out difference in the four-seed study. Its central advantage over post-rollout filtering is avoiding suffix generation. Calibrated task-specific signals and continued audits of false cuts are necessary for wider use.

Acknowledgments and Disclosure of Funding

This work was supported by the Innovation Program of Shanghai Municipal Science and Technology Commission under Grant Nos. 26530713400 and 25DP1500300, and the National Natural Science Foundation of China under Grant No. 62231010. The authors declare no competing interests.

References

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

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

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

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

[5] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[6] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, 2022.

[7] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations (ICLR)*, 2024.

[8] 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.

[9] 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. *arXiv preprint arXiv:2506.02177*, 2025.

[10] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[11] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[12] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[13] Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, second edition, 2018.

[17] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 12248–12267, 2024.

[18] Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *International Conference on Machine Learning (ICML)*, 2024.

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

[20] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

[21] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

[22] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, 2020.

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

[24] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, John Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[25] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *International Conference on Machine Learning (ICML)*, 2024.

[26] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 41–48, 2009.

[27] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *International Conference on Machine Learning (ICML)*, pages 1311–1320, 2017.

[28] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022.

[29] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, pages 19274–19286, 2023.

[30] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[31] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

[32] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[33] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[34] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[35] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

[36] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

[37] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

[38] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations (ICLR)*, 2024.

[39] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

[40] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[41] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[42] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*, 2022.

[43] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

[44] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[45] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[46] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

[47] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

[49] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[50] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2023.

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

[52] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[53] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[54] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning (ICML)*, 2024.

[55] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[56] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[57] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

A Related work

Group-relative policy optimisation and its variants. GRPO [1, 2] replaces the learned value baseline of TRPO [15] / PPO [4] with the empirical mean of G rollouts of the same prompt; both descend from REINFORCE-style policy gradients [10, 16]. Several follow-up variants modify the advantage computation: RLOO [17] uses a leave-one-out estimator, ReMax [18] uses the maximum reward as the baseline, Dr.GRPO [19] removes the σ_r normalisation, and REINFORCE++ [20] adds token-level normalisation. None of these addresses the zero-variance group problem directly: when all G rewards are equal, any baseline based on the group itself yields zero advantage. They change the gradient magnitude on non-zero-variance groups but leave the wasted rollout cost on zero-variance groups untouched. The broader RLHF lineage [21, 22, 3], including DPO [23], Constitutional AI [24] and RLAIF [25], shares the policy-gradient backbone but optimises against learned reward or preference models rather than environment-grounded rewards.

Filtering convergent groups: prompt-level and in-rollout. DAPO [8] drops zero-variance groups from the gradient batch after the rollouts have already finished, replacing them with new prompts to keep the batch full; exact but post-hoc. GRESO [9] predicts which prompts will be zero-variance before any rollout begins, using cross-epoch reward consistency on mathematical reasoning benchmarks. Both decide at the prompt level and are orthogonal to a mid-rollout decision. Curriculum-learning approaches [26, 27] also reweight prompts at the prompt level, by difficulty rather than by predicted variance. We are not aware of prior work that uses intermediate rollout state to predict zero-variance groups in GRPO. The closest single-trajectory analogues are early-exit inference: CALM [28] terminates a single generation when its per-token confidence saturates, and speculative decoding [29, 30] verifies a fast draft model against the target. These act on one trajectory; our setting requires a decision over a group of G trajectories using their disagreement as the signal.

LLM agents and agent RL benchmarks. ALFWorld [5], WebShop [6], WebArena [31], and AgentBench [7] are standard testbeds for LLM agent behaviour, with WebGPT [32] an early instance of agent RL on browser tasks. They share a structural property that mathematical reasoning benchmarks lack: every rollout is a multi-turn trajectory through an observable state space, so the shape of an in-progress group can be inspected at any intermediate step. ReAct [33] interleaves reasoning with action, Reflexion [34] and Self-Refine [35] add verbal self-critique, Tree-of-Thoughts [36] explores multiple reasoning branches, and Toolformer [37] / ToolLLM [38] expand the action space with API calls. Voyager [39] embeds an LLM agent in an open-ended environment with skill acquisition. These contributions design *what* the policy does inside a rollout; ours is about *when to stop* the rollout.

Multi-sample inference and training efficiency. The $G\times$ multi-sample structure also appears at inference time: self-consistency [40] takes a majority vote over G chain-of-thought [41] rollouts, rationale-augmented ensembles [42] aggregate weighted samples, best-of-N sampling with a verifier or reward model picks the best of N candidates [43, 44], and pass@k [45] is the standard G-sample capacity-evaluation metric. The same divergence signal applies to these settings. A complementary line of work reuses trajectories for self-training rather than discarding them: STaR [46], ReST [47] and *Beyond human data* [48] iterate on model-generated solutions filtered by reward, which *exploits* successful rollouts where our gate *cuts* saturated ones. Orthogonal lines of work reduce per-token cost via parameter-efficient fine-tuning (LoRA [49], QLoRA [50]) and inference-system optimisations (PagedAttention [51], FlashAttention [52, 53], Medusa [54]); our gate reduces the *number* of steps generated rather than the cost of each step. Throughout we use Qwen2.5-7B-Instruct [11] as the base policy, in line with current open-base-model practice [55, 56].

B Full mid-rollout signal definitions

The body (§3.2) defines d_K and points here for the other six in-group divergence signals we considered. Each is a scalar in $[0, 1]$ computed from the G partial trajectories. Edit/Jaccard distance and normalized entropy are zero for identical prefixes or actions; unique ratios have minimum $1/G$. Termination fraction measures completion, not divergence.

- **prefix_edit_distance_mean** (d_K). The body signal: mean over $\binom{G}{2}$ pairs of the Levenshtein edit distance between $a_{i,1:K}$ and $a_{j,1:K}$, divided by the longer length.

- **action_bigram_jaccard_mean**. For each pair of trajectories, take the set of consecutive action pairs (action bigrams) in each trajectory's first K steps, compute the Jaccard overlap of the two sets, and report 1 minus that overlap. Average over all pairs. Like d_K this is 0 when the trajectories use the same action transitions and grows as they diverge.
- **unique_prefix_ratio**. Number of distinct trajectories among $a_{i,1:K}$, divided by G. The minimum $1/G$ means all G trajectories share the same first-K sequence; the maximum 1 means every trajectory is different. **unique_action_ratio** is the same quantity computed on the single action $a_{i,K}$ at step K rather than on the full prefix.
- **action_entropy**. Shannon entropy of the empirical distribution of $\{a_{i,K}\}$ across the G trajectories at step K, divided by $\log_2 G$ to normalize it to $[0, 1]$.
- **obs_unique_ratio**. Same as **unique_action_ratio** but on the environment observation $o_{i,K}$ rather than the action; a proxy for whether the trajectories have reached the same world state.
- **termination_fraction** (τ_K). Fraction of the G trajectories with fewer than K recorded actions in the released implementation (completed before the checkpoint action). $\tau_K = 0$ means none has finished and $\tau_K = 1$ means all have. It lies in $[0, 1]$ but is not a divergence measure; we consider it as a complementary signal in App. E.9.

The full 7×4 side-by-side comparison (Spearman ρ and AUROC for each metric at $K \in \{5, 10, 15, 20\}$) is in App. E.3.

C Implementation details

C.1 How the gate is enforced inside a rollout

The rollout loop generates the G trajectories of a group one environment step at a time. After the K-th environment step we pause the loop and compute the in-group divergence d_K on the partial trajectories produced so far. If d_K falls below the threshold d_L, every trajectory in the group is recorded as terminated at step K with whatever reward it has accumulated by then, and the rollout loop exits early for that group. The remaining $T_{\max} - K$ steps are not generated. **The accumulated step-K rewards are kept on the trajectory records only for logging and visualization** (e.g. the training-reward panel of Fig. 4 subtracts cut groups precisely because their truncated reward of 0 is not a real outcome); **the cut group itself is dropped from the GRPO advantage and loss computation entirely** (§3.2, line '`if is_cut: continue`' in the released code), so these truncated rewards never enter the gradient. Independent groups may be scheduled on different GPUs, but trajectories within a group require a shared checkpoint decision. Parallel wall-clock saving depends on scheduling and stragglers, not simply the sum of per-group savings. No model parameters are changed during this rollout phase; the gate only affects how much of each group's trajectories is generated.

C.2 Teacher-forcing GRPO loss in agent setting

After rollout, each trajectory $\boldsymbol{\tau}$ is a token sequence $[\text{prompt}, o_1, a_1, o_2, a_2, \ldots, o_T, a_T]$ where o_t is the environment observation at step t and a_t is the model's response. Only a_t tokens are produced by π_θ; o_t tokens are environment-injected. Computing the policy-gradient loss requires $\log p_\theta(a_t \mid \text{prefix})$ at every action-token position.

The implementation samples environment-step items. Each item consists of the interaction prefix followed by one assistant response, including its `Thought:` and `Action:` text. A teacher-forcing forward pass obtains the response-token log-probabilities; a mask constructed from the prefix length excludes prompt and observation tokens. The item loss is the negative group-relative advantage times the mean response-token log-probability, averaged over selected items. Tier 3 uses a cap of 96 items per iteration. Thus the implemented objective is a sampled response-level surrogate, rather than a full-trajectory log-likelihood sum.

C.3 Reloading the policy between iterations

The on-policy training run in Tier 3 alternates between sampling rollouts and updating the policy. After each gradient update, the trained LoRA adapter weights are written to disk and then loaded back into the inference engine before the next iteration's rollouts begin. The base model itself stays

resident in GPU memory and is not reloaded; only the small LoRA delta is swapped. The total round-trip cost is on the order of one second per iteration at LoRA rank 16, which is negligible compared to the time spent generating the iteration's rollouts.

C.4 Hyperparameters

Table 6 lists the hyperparameters used in each of the three integration tiers. The same gate setting $(K, d_L) = (10, 0.12)$ is used in all three tiers, chosen on the offline buffer in §3.5 and not retuned for the online runs. Tier 2 additionally uses the OR-rule backup $\tau_K \geq 0.90$. Tier 1 has no training so the LoRA and optimiser hyperparameters do not apply. Tier 2 uses a smaller LoRA rank ($r = 8$, two target modules) because the buffer is fixed and the optimisation is short (20 steps); Tier 3 uses a larger LoRA rank ($r = 16$, seven target modules) because the policy must continue to learn over 60 on-policy iterations. The sampling temperature during rollout is 0.7 in all tiers, which keeps the G trajectories of a group meaningfully different from each other; held-out evaluation in Tier 3 uses greedy decoding (temperature 0) so each test task is decoded once and the outcome is deterministic.

Table 6: Hyperparameters across the three integration tiers.

	Tier 1 (rollout-only)	Tier 2 (off-policy)	Tier 3 (on-policy)
prompts	100 (baseline & gated, same seed)	100 (replay buffer)	60×10 resampled
gradient steps	0	20	60
groups per step	—	4	10
training items per update	—	up to 128 (recorded)	up to 96
LoRA rank / α	—	8 / 16	16 / 32
LoRA targets	—	q, v	q, k, v, o, MLP
optimizer	—	AdamW	AdamW
learning rate	—	5×10^{-5}	5×10^{-5}
sample temp. (rollout)	0.7	0.7	0.7
sample temp. (eval)	—	—	0 (greedy)
$T_{\max}$	30	30	30
G	8	8	8
gate (K, d_L)	$(10, 0.12)$	$(10, 0.12)$	$(10, 0.12)$
random seed	42	42	$\{7, 13, 23, 42\}$

C.5 Setting the precision floor

The 0.80 precision floor is an empirical calibration choice. At $(K, d_L) = (10, 0.12)$ there are 21 cuts, including 4 mixed groups: precision is $17/21 = 0.810$ and the mixed-group false-cut rate is $4/61 = 6.56\%$. A task-group bootstrap gives precision 95% CI $[0.625, 0.957]$. This does not imply a population precision guarantee or a bound on policy-gradient norm. The 96.7% advantage-vector norm retained in the completed buffer is a separate empirical diagnostic; it does not include score-function vectors or optimizer effects.

C.6 Hardware and timing

The original ALFWorld experiments use NVIDIA RTX 6000 Ada (48 GB) GPUs. Tier 1 schedules the 100 tasks across up to 4 GPUs. Tier 3 uses two GPUs, one for sampling and one for gradient updates. Each on-policy training run takes approximately 4.1–4.9 hours. The eight Tier-3 runs alone imply about 73 allocated GPU-hours at two GPUs per run; this is an allocation estimate, not a measured device-utilization total. A complete project-wide GPU-hour accounting is unavailable. Inference uses vLLM [51] and gradient updates use the Hugging Face Transformers [57] and PEFT/LoRA [49] stacks.

D Extended discussion

D.1 Normalization effects and their limits

For fixed per-trajectory contributions $h_i = A_i \nabla_\theta \log p_\theta(\tau_i)$, deleting exactly zero contributions leaves $\sum_i h_i$ unchanged. Replacing denominator N by N_{kept} rescales that sum by N/N_{kept}. This

identity does not hold for the numerator when mixed groups are falsely stopped, and it does not establish an unbiased estimator under random normalization.

The prompt-sampling distribution can stay uniform while the distribution of groups entering an update changes. Neither this selection effect nor the role of optimizer state is excluded by our experiment. The gradient-norm ratio is descriptive evidence; a learning-rate-matched or denominator-matched ablation would be needed to isolate its effect on held-out performance.

D.2 Optional randomized continuation: a conditional identity

The deterministic ALFWorld gate has no unbiasedness guarantee. For a fixed proposed batch and fixed parameter value, let g_{full} be the gradient computed after all groups finish, and g_{stop} the gradient with the proposed groups dropped. Independently draw $Z \sim \text{Bernoulli}(q)$, $0 < q \leq 1$, before observing the missing suffixes. If $Z = 1$, complete the proposed groups and compute both gradients with their explicitly specified normalizations. Then

$$\widehat{g} = g_{\text{stop}} + \frac{Z}{q}(g_{\text{full}} - g_{\text{stop}}), \qquad \mathbb{E}_Z[\widehat{g} \mid g_{\text{full}}, g_{\text{stop}}] = g_{\text{full}}. \tag{3}$$

The equality follows from $\mathbb{E}[Z/q] = 1$, assuming continuation does not alter the potential full trajectories. Expected suffix saving is $(1 - q)$ of deterministic stopping; additional gradient-estimator variance grows as $(1 - q)/q$. Groupwise weights Z_i/q_i give the analogous result only for additive contributions with a fixed denominator; simply weighting advantages does not generally correct a random denominator. The batch-level identity does not imply unbiased nonlinear optimizer updates or identical learned policies. We report it as an optional correction, not as a property demonstrated by the original ALFWorld experiments.

D.3 Future directions

Environment-grounded progress signals for high-d_K all-failure groups. The gate catches the low-d_K end of the all-fail cluster ($10/25$ in our buffer; §3.4) but cannot reach the high-d_K majority because their early-action prefixes overlap with mixed-outcome groups in the same d_K region. Detecting them would require a signal beyond what the partial trajectories themselves provide: for example, an intermediate progress proxy from the environment ("object picked up", "sub-goal reached") that distinguishes a stuck rollout from one that is still exploring. ALFWorld and most agent benchmarks expose such proxies, but using them is outside the scope of the main experiments.

Per-group choice of the gate evaluation step. The main gate uses a fixed $K = 10$. Some groups show predictive prefix convergence earlier (by $K = 5$); others are not separable until $K = 15$. A model that predicts the optimal K for each group from a few early steps of the rollout could recover further compute.

Composition with DAPO and GRESO. Our gate is orthogonal to both filters. GRESO can skip a prompt before any rollout begins, our gate can stop a group's rollout partway through, and DAPO can drop a group from the gradient batch after the rollout has finished. The three operate at three distinct stages of the training pipeline and compose naturally. A natural follow-up is to measure the combined saving end-to-end.

E Detailed experimental results

E.1 Calibration against random-cut and oracle gates

We compare with two reference policies on the same 100 complete groups. A uniformly random cut of exactly 21 groups matches R1's cut budget. It has expected 8.19 true and 12.81 false positives, and expected advantage-L^2 retention of 88.9%, compared with R1's 96.7%. The random quantities are expectations, not one sampled realization.

The oracle cuts all 39 zero-variance groups at $K = 10$. Actual-length savings sum $\max(T_i - K, 0)$ over trajectories and divide by the $18{,}197$ total executed actions. The oracle saves $4{,}200$ actions (23.08%); R1 saves $2{,}170$ gross (11.93%), of which $1{,}723$ (9.47%) come from true positives. R1

therefore recovers 41.0% of the oracle's lossless action savings. This accounting differs from the fixed-horizon proxy in Table 1.

E.2 Bootstrap confidence intervals on the headline savings

The Tier 1 A/B run (§4.1, Table 3) measures the gate's effect on summed per-task durations: the gated total is 13.25% lower than the baseline on the same 100 ALFWorld tasks. To check that this saving is robust to the particular set of 100 tasks we drew, we resample the 100 paired task timings with replacement $10{,}000$ times (NumPy generator seed 42) and recompute $1 - \sum t_g / \sum t_b$. The percentile 95% confidence interval is $[7.47, 19.03]\%$; the interval excludes zero.

The offline fixed-horizon proxy is distinct from the timing result. For the 100 complete groups, the audited actual-length savings are 11.93% gross (95% task-group bootstrap CI $[6.83, 17.58]\%$) and 9.47% from true-positive cuts (CI $[4.72, 14.80]\%$), using $10{,}000$ resamples. Neither interval is a prediction interval for wall-clock time, which depends on hardware and scheduling.

E.3 Full correlation table

Table 7: Spearman ρ of each (metric, K) cell against group reward variance σ_r^2, $N = 100$, $G = 8$. Cells in **bold** exceed the descriptive highlighting threshold $|\rho| \geq 0.40$. Termination fraction has near-zero monotone correlation by design (U-shape against label).

metric	$K=5$	$K=10$	$K=15$	$K=20$
prefix_edit_distance_mean (d_K)	0.246	0.374	**0.419**	**0.418**
action_bigram_jaccard_mean	0.293	**0.407**	**0.406**	0.389
unique_prefix_ratio	0.320	0.398	0.307	0.208
unique_action_ratio	0.314	0.379	0.272	0.167
action_entropy	0.292	0.369	0.255	0.161
obs_unique_ratio	0.251	0.397	0.307	0.093
termination_fraction (τ_K)	-0.188	-0.059	0.110	0.235

E.4 False-positive case analysis

At $d_L = 0.12$ the gate incorrectly cuts 4 mixed groups, all `pick_and_place_simple` tasks. Their prefixes contain similar receptacle-search actions (three drawer cases and one cabinet case), but are not identical across every trajectory. Later exploration produces both successes and failures. Low prefix divergence therefore does not establish identical future outcomes; these cases motivate continued false-cut audits after calibration.

E.5 Per-task-type correlation breakdown

Table 8 reports the gate signal for each of the six ALFWorld task types, in two forms: the global metric d_{10} that the gate actually uses, and the post-hoc best $(metric, K)$ pair per type. The global d_{10} generalises unevenly: it is excellent on `look_at_obj_in_light`, `pick_heat`, and `pick_two_obj` (AUROC ≥ 0.91) but degrades on `pick_and_place_simple`, `pick_clean`, and `pick_cool` (AUROC 0.48–0.62). Allowing the $(metric, K)$ pair to be tuned per type recovers the worst three to AUROC 0.74–0.86, with `termination_fraction` at $K = 20$ being the best signal in the harder pick-cool / pick-clean tasks. The two task types where the per-type best is AUROC 1.00 have only $n_{zv} = 1$–2 zero-variance groups, so those cells are essentially single-observation; they should be read as "the gate did not fail" rather than "perfect".

E.6 $G = 8$ vs $G = 16$ robustness panel

The whole paper uses a group size of $G = 8$. To check that our findings still hold at a larger group size, we re-collected the $N = 100$ ALFWorld rollouts with $G = 16$ trajectories per prompt and recomputed the divergence measures and correlations.

Two things stay essentially unchanged. The group-label composition is similar: 25 all-fail, 61 mixed, 14 all-succeed at $G = 8$ vs 25 all-fail, 62 mixed, 13 all-succeed at $G = 16$ (39 vs 38 zero-variance

Table 8: Per-task-type AUROC for the (binary) zero-variance vs non-zero-variance classification at the published gate's evaluation step $K=10$. **Global d_{10} AUROC** is the metric the gate actually uses; **Best AUROC** is the maximum over $(metric, K)$ tuned per task type, with the winning pair in parentheses. n = groups in that type; n_{zv} = zero-variance groups; the positive class for AUROC is non-zero variance. Metric names are abbreviated.

Task type	n	n_{zv}	Global d_{10} AUROC	Best $(metric, K)$	Best AUROC
`pick_and_place_simple`	24	13	0.62	(unique action, 15)	0.86
`pick_two_obj_and_place`	20	13	0.91	(bigram Jaccard, 10)	0.92
`pick_cool_then_place`	19	5	0.48	(termination, 20)	0.82
`pick_clean_then_place`	18	5	0.56	(termination, 20)	0.74
`pick_heat_then_place`	11	1	1.00*	(action entropy, 10)	1.00*
`look_at_obj_in_light`	8	2	1.00*	(unique prefix, 15)	1.00*
median	—	—	0.76	—	0.89

* $n_{zv} \leq 2$, single-observation regime.

groups out of 100). The shape of the correlation curve in K is also similar (Figure 5): the divergence-variance correlation rises from $K = 5$, peaks around $K = 10$–15, and is roughly flat or slightly decreasing afterwards, in both group sizes.

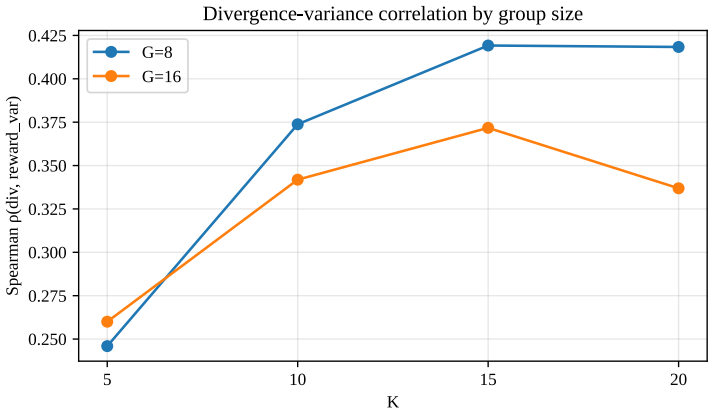

Figure 5: Spearman ρ between mid-rollout divergence d_K and final reward variance σ_r^2, plotted as a function of the gate evaluation step K, for two group sizes ($G=8$ and $G=16$) on the same 100 ALFWorld tasks. The shape of the curve is the same in both group sizes; the peak value is slightly lower at $G=16$ (0.37 vs 0.42), which we explain in the text.

The peak correlation is slightly lower at $G=16$ ($\rho=0.37$ at $K=15$) than at $G=8$ ($\rho=0.42$). The reason is mechanical: at $G=8$ a converged group (whether heading to a common success or stuck in a common failure) often has all eight trajectories choose the same first K actions, giving $d_K \approx 0$ and a clear separation from non-zero-variance groups; at $G=16$ it is more common that at least one of the sixteen trajectories takes a slightly different first action, so d_K is no longer near zero even when the group ends up zero-variance. This thins the low-d_K zero-variance cluster (both all-succeed and all-fail) a little and reduces the rank correlation.

Threshold transfer is partial. Applying the single-axis gate $(K = 10, d_L = 0.12)$ verbatim to the $G=16$ data flags more groups but at lower precision, because with more samples per group it is more likely that a non-zero-variance group happens to share a low-divergence prefix by chance. Re-fitting the threshold on $G = 16$ at the same 0.80 precision floor moves the operating point to $(K=15, d_L=0.09)$, with savings of about 7.0%, strictly lower than at $G=8$ because each cut group at $K = 15$ recovers only $T_{\max} - K = 15$ remaining steps instead of 20. The gate transfers across group sizes, but its threshold should be retuned at each G.

E.7 Off-policy training per-step trace

In Tier 2 each training step samples 4 groups from the buffer. A training item is a selected (trajectory, action-position) pair. The recorded baseline uses 128 items per step; the gated arm uses 64–128. The archived trace does not establish the exact per-trajectory sampling cap, so we report the observed item counts.

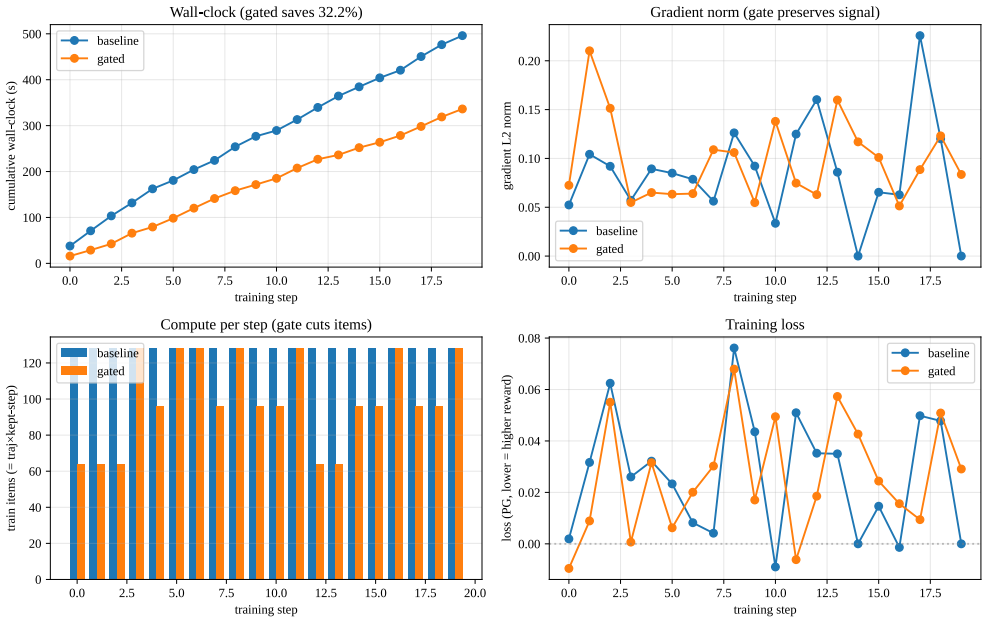

Figure 6: 20-step off-policy GRPO training A/B on Qwen2.5-7B + LoRA. *Top left:* cumulative wall-clock; the gated arm finishes 32.2% faster in the fixed-buffer training phase. *Top right:* per-step gradient L^2-norm; the gated arm runs at about 114% of baseline. *Bottom left:* number of training items per step (one item = one trajectory × one action position; max 128 as explained above). The baseline arm always uses the maximum 128 items because no group is cut. The gated arm drops to between 64 and 128 items per step depending on how many groups the gate cut at that step. *Bottom right:* per-step training loss. Both arms sit in the same noisy band around zero. Zero-sum advantages do not imply a zero expected policy-gradient loss because log-probabilities are not constant. This loss is not a direct reward measure; held-out evaluation in Tier 3 measures policy quality.

E.8 On-policy training per-iter trace and held-out trajectory

In Tier 3, every 10 iterations we pause training and run greedy decoding on 50 held-out ALFWorld tasks (§4.3). Table 9 shows the resulting success rate at each of the 7 evaluation points (iters $0, 10, 20, \ldots, 60$), averaged over the 4 random seeds we ran. At iter 0 no training has happened yet, so the gated and baseline arms share the same policy by construction and therefore evaluate identically per seed (the across-seed std is non-zero because the held-out task set is seeded from the run's random seed). The held-out gap between the two arms is non-monotonic across iterations: gated is slightly behind at iters 20–30 and slightly ahead from iter 40 onward. The +2.5 pp mean gap we report at iter 60 in the main text is one point on this noisy trajectory.

Per-iter gate firing pattern. The per-iter cut count is highly variable, ranging from 0 to 6 groups (out of the 10 in each iteration) depending on the difficulty of the iteration's sampled prompts. Across the 4 seeds and 60 iterations the gate fires on a mean of $\mathbf{106 \pm 18}$ groups out of 600 total (17.7% cut rate; per-seed totals $88, 94, 119, 124$ for seeds $13, 7, 23, 42$). The mean baseline zero-variance rate over the same iters is 40.6%, and the cut rate cannot be interpreted as recall: the two arms use different sampled trajectories and eventually different policies. Completed counterfactuals would be needed to identify false cuts online.

Table 9: Held-out success rate across the 60 on-policy iterations, averaged over the 4 seeds. Δ is the per-seed mean of (gated − baseline). At iter 0, gated and baseline are the same policy (same random seed), so per-seed they coincide and $\Delta = 0$ exactly. Across-seed standard deviations are reported in Figure 4 (shaded bands) and Table 5 (iter 0 and iter 60).

iter	0	10	20	30	40	50	60
baseline	39.5%	38.5%	39.5%	40.5%	39.0%	37.0%	41.5%
gated	39.5%	39.5%	38.0%	37.5%	43.5%	40.0%	44.0%
Δ	0	+1.0	−1.5	−3.0	+4.5	+3.0	+2.5

Has the policy converged? The training reward variance does not collapse over the 60 iterations in either arm, so neither policy has reached a fixed point within the compute budget we used. Longer training could change both the wall-clock saving and the held-out gap as the policy and trajectory-length distribution evolve. The $n = 4$ seeds and 60 iterations cannot determine longer-run behavior.

E.9 The OR-rule extension: optional termination-fraction backup

The main paper uses a one-parameter single-axis gate $\textsc{Cut}_{K,d_L}$: `cut iff` $d_K < d_L$ (§3.2). We document here a two-parameter *OR-rule* extension we explored:

$$\textsc{Cut}^{\vee}_{K,d_L,\tau_H} : \quad d_K < d_L \ \vee \ \tau_K \geq \tau_H, \tag{4}$$

which adds a backup clause that triggers when most of the group has already terminated. The motivation is the U-shape between τ_K and the group label (Table 10 below): all-success groups concentrate at high τ_K *and* low d_K; high τ_K contains a few extra all-success groups whose trajectories arrived at the goal via slightly different short prefixes that the d-threshold misses.

Table 10: Median of d_K and τ_K at $K = 15$ stratified by group label, $G = 8$. The two zero-variance classes (all-fail, all-succeed) sit on *opposite* ends of τ_K, motivating the U-shape interpretation that originally led us to the OR-rule.

group label	n	median d_{15}	median τ_{15}	physical meaning
all_succeed	14	0.19	1.00	solved fast, all terminated
mixed	61	0.47	0.13	exploring different strategies
all_fail	25	0.26	0.00	nobody finishes, often lock-stepped

Quantitative effect of adding the τ backup. At $K = 10$, $\tau_H = 0.90$, $d_L = 0.12$, the OR-rule extension flags 23 groups (vs 21 for $d_K < 0.12$ alone) and hits all 2 extra groups as true positives, yielding precision 0.83 vs 0.81 and recall 0.49 vs 0.44. **However, the incremental rollout savings is exactly zero**: by step $K = 10$ those 2 extra groups have $\tau \geq 0.90$ — i.e., all 8 of their $G = 8$ trajectories have *already* terminated, so "cutting" them at K saves nothing the early-termination mechanism didn't already save (Table 12).

We therefore recommend the single-axis gate as the production default: simpler, less prone to overfitting on small validation sets ($N = 100$ here), and indistinguishable in rollout savings. The OR-rule is documented for completeness and would be the natural extension on environments where τ_K at moderate K is more discriminative than ALFWorld provides.

Trying to catch all-failure groups. The OR-rule above adds a backup clause for the all-success end of the U-shape ($\tau_K \geq 0.90$). The natural mirror, which would catch all-failure groups, is to add a backup clause for the low-τ end: $d_K < d_L \ \vee \ \tau_K \leq t_L$. We swept this rule across $d_L \in [0.02, 0.30]$, $t_L \in \{0, 0.05, 0.10, 0.15\}$, and $K \in \{5, 10, 15, 20\}$. *No operating point clears the* 0.80 *precision floor at any* K. The reason is geometric: in early rollout phases mixed groups also have $\tau_K \approx 0$ (some trajectories are still running, none has reached a terminal state yet), so any cut on low τ_K pulls many mixed groups in along with the all-failure ones it was meant to catch. This result concerns the low-termination backup only, not the high-termination OR rule above.

E.10 OR-rule training A/B (seed 42, R3)

Before pivoting to the single-axis R1 gate, we ran the on-policy training A/B with the two-parameter OR-rule extension R3 (App. E.9) on seed 42. We report those numbers here for completeness:

Table 11: On-policy A/B with the OR-rule extension R3 ($d_K < 0.12 \vee \tau_K \geq 0.90$), seed 42, 60 iters. Setup otherwise matches Tier 3 in the main paper.

	baseline	gated (R3)	Δ
total wall-clock (s)	17,252	15,364	-10.94%
groups cut / total	$0/600$	$96/600$	cut rate 16.0%
gradient L^2-norm mean	0.123	0.181	$+47\%$
held-out eval, iter $0/60$	$38.0\%/38.0\%$	$\mathbf{38.0\%/46.0\%}$	$+\mathbf{8}$ pp

On this single seed (seed 42), the OR-rule extension gives a $+8$ pp held-out improvement over its baseline. The single-axis gate on the same seed gives $+2$ pp, and across the four single-axis seeds we report in the main paper (seeds $\{7, 13, 23, 42\}$) the held-out improvement is $+2.5 \pm 3.4$ pp on average. The OR-rule result is one seed, with no reliable paired confidence interval available from the archived aggregate outcomes. It does not establish a policy-quality advantage over the single-axis rule, nor a causal effect of gradient normalization. Tier 2 also uses the OR backup; the four-seed Tier-3 result uses R1.

E.11 Strong-baseline comparison detail

Table 12: Same-buffer comparison at $K = 10$. Savings use actual remaining actions, not the fixed-horizon proxy. Raw includes all cuts; safe includes only true positives. Random-cut entries are exact-budget expectations. The DAPO rows model only post-rollout zero-variance filtering, not a trained DAPO baseline. L^2 denotes advantage-vector norm retention.

arm	cut	TP	FP	prec.	raw (%)	safe (%)	L^2 (%)
no gate	0	0	0	—	0.0	0.0	100.0
random (matched 21)	21	8.19	12.81	0.39	12.36	4.85	88.9
oracle	39	39	0	1.00	23.08	23.08	100.0
post-rollout filter	0	0	0	—	0.0	0.0	100.0
OURS R1	21	17	4	0.81	11.93	9.47	96.7
$\tau_K \geq 0.90$ only	6	6	0	1.00	0.0	0.0	100.0
OR extension R3	23	19	4	0.83	11.93	9.47	96.7
R1 + post-rollout filter	21	17	4	0.81	11.93	9.47	96.7

Post-rollout filtering saves no suffix generation because rewards must first be observed. Combining it with R1 retains the same rollout savings while removing remaining zero-variance groups before training. Its full training cost is not measured by this offline comparison; Tier 2's 32.2% saving comes from a separate OR-gated diagnostic.

E.12 Mechanism evolution: group rates and gradient norms

The baseline zero-variance group fraction averages 40.6% across 240 seed-iteration records. The archived gated proxy, $\max(n_{\text{zv}} - n_{\text{cut}}, 0)/(n_{\text{groups}} - n_{\text{cut}})$, averages 36.6%. At seed 13, iteration 24, $n_{\text{zv}} = 3$ but $n_{\text{cut}} = 4$; this record contributes zero to the proxy. These group-count summaries cannot establish token-weighted batch dilution, online precision, or recall. Mean gradient norms are 0.125 and 0.145 for baseline and gated arms, respectively. Their ratio is descriptive and does not establish the cause of the held-out gap.

F Prompt templates

ALFWorld system prompt. We use a ReAct-style prompt format. The system message describes the ALFWorld environment, lists the canonical action vocabulary, and provides one in-context example trajectory. The user message at each step contains the current observation, and the assistant

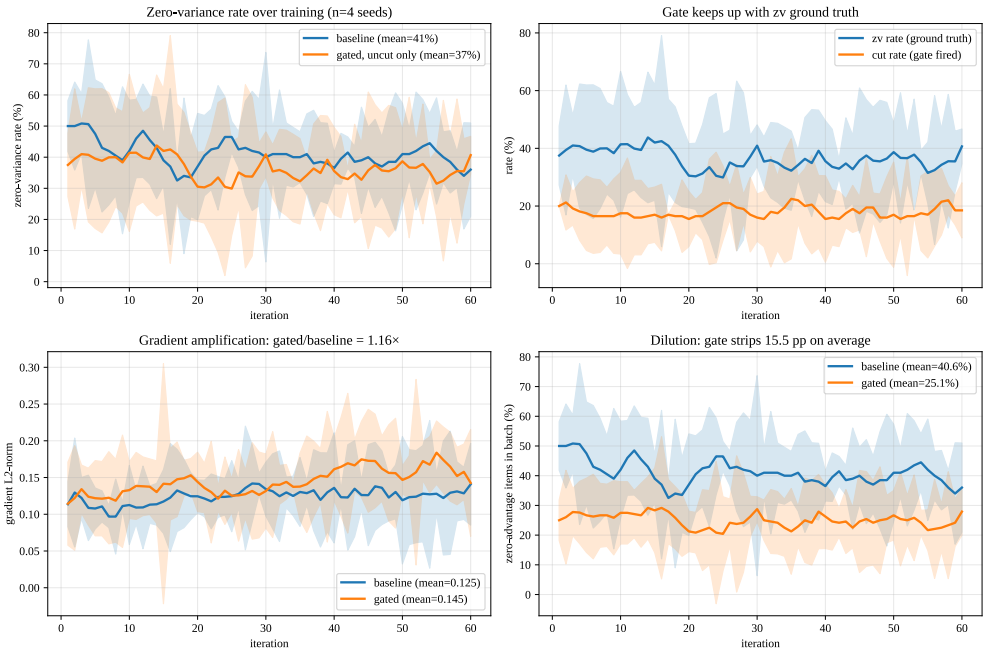

Figure 7: On-policy diagnostics ($n = 4$ seeds, 60 iterations). Lines are five-iteration rolling means; bands use per-iteration cross-seed standard deviations. Top-left and top-right: gated zero-variance curves are log-derived proxies, not full-return ground truth; their comparison with cut rates does not estimate recall. Bottom-left: recorded gradient norms (ratio $1.16\times$), without a causal interpretation. Bottom-right: the original 25.1% and 15.5 pp annotations use inconsistent normalizations across arms and are not valid estimates of training-batch dilution.

is expected to emit text in the format "`Thought: ... Action: <action>`". The action string is parsed greedily; if it does not match the environment's action grammar, the step is treated as a no-op and still counts towards $T_{\max}$.

Eval prompt. Held-out eval (Tier 3) uses the same prompt format but with sampling temperature $T = 0$ (greedy) and no gate. A task is counted as solved iff the environment returns $r = 1$ within $T_{\max} = 30$ steps.

G Broader impacts

The selective-rollout gate reduces the compute footprint of agent RL training in the evaluated ALFWorld runs: 13.25% rollout-only, 32.2% fixed-buffer training, and 10.7% on-policy elapsed time. The held-out difference is not statistically significant. Lower runtime can reduce the resource cost of LLM agent training, which aligns with the broader goal of making such training accessible to research groups without hyperscale compute. We do not foresee novel dual-use risks beyond those already inherent to LLM agent RL: any compute-saving technique applied to capability research generally accelerates capability research, including capabilities that may have negative externalities (e.g. autonomous web agents executing harmful actions). The gate itself is agnostic to the reward function and does not introduce new incentives.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (§1) state three empirical claims — (i) a mid-rollout gate that recovers a measurable fraction of rollout compute, (ii) a precision/recall trade-off measured on complete offline groups, and (iii) end-to-end wall-clock savings with a positive but statistically inconclusive held-out difference — supported by the experiments in §3–§4. The gate does not guarantee unbiased gradients or improved policy quality.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated "Limitations" paragraph in §5 and the analysis in App. E.9 discuss (i) failure to detect the high-divergence tail of convergent-failure groups, (ii) adding a low-termination backup did not meet the precision floor in the explored sweep, (iii) only $n=4$ seeds for the on-policy run so the held-out improvement is directional but not statistically significant, (iv) fixed model and environment (Qwen2.5-7B + ALFWorld), and (v) the gate evaluation step K is a fixed hyperparameter rather than chosen adaptively.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The contribution is primarily empirical. The compute decomposition in §2 and the precision-floor derivation in §C.5 state their assumptions; the optional conditional identity in §D.2 gives its assumptions and one-line derivation, without claiming an unbiased learned policy or optimizer update.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: §4 describes the three-tier protocol, §C gives the gate enforcement loop, the teacher-forcing GRPO loss, and the LoRA hot-swap procedure, Table 6 lists the experimental hyperparameters (including the four seeds), §C.6 reports the hardware, and §F gives the prompt templates. The public repository provides implementation code and result tables, but does not include the complete raw rollout buffers or training logs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Implementation code and result tables are available at `https://github.com/zhiyuanZhai20/selective-rollout`, with entry points described in the README. The complete raw rollout buffers and training logs are not included, so the public release alone does not support exact reproduction of every reported result.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Table 6 in §C.4 lists optimizer, learning rate, LoRA rank/α/targets, sampling temperature, G, $T_{\max}$, K, d_L, and seeds for the three tiers. The train/held-out split and checkpoint threshold ($K = 10, d_L = 0.12$) are described in §4.3 and §3.5; the archived Tier-2 implementation additionally uses a termination-fraction backup clause, as disclosed in the experimental protocol.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The Tier-1 saving in summed per-task elapsed time is reported with a 10,000-resample paired task-bootstrap 95% CI of $[7.47, 19.03]\%$ (seed 42; §E.2). The Tier-3 numbers in Table 5 and Fig. 4 are mean$\pm$std across $n = 4$ random seeds $\{7, 13, 23, 42\}$, with per-seed values disclosed; the held-out $+2.5$ pp gap is directionally positive but not statistically significant at this seed count.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: §C.6 reports NVIDIA RTX 6000 Ada (48 GB) GPUs, with up to 4 GPUs for Tier 1 and 2 GPUs for each on-policy run, taking 4.1–4.9 hours per on-policy run. The eight Tier-3 runs imply approximately 73 allocated GPU-hours; a complete project-wide accounting, including preliminary and discarded runs, is unavailable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We have read the NeurIPS Code of Ethics. The work uses only public model weights (Qwen2.5-7B-Instruct) and a public synthetic-text simulator (ALFWorld); it does not involve human subjects, personal data, or scraped web data, and we have preserved anonymity throughout the submission.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: §G discusses the positive impact (lower energy and GPU-hour cost of agent RL training, broadening access for groups without hyperscale compute) and the negative impact (any compute-saving technique applied to capability research generally accelerates capability research, including potentially harmful agent capabilities). The gate is reward-agnostic and does not introduce new misuse incentives beyond those inherent to LLM agent RL.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We do not release any new pre-trained language model, image generator, or scraped dataset. The public release consists of implementation code and result tables for experiments in the public ALFWorld simulator with public Qwen2.5-7B weights; it does not introduce a new high-risk data or model release.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Qwen2.5-7B-Instruct (Apache 2.0), ALFWorld (MIT), LoRA / PEFT (Apache 2.0), Hugging Face Transformers (Apache 2.0), and vLLM (Apache 2.0) are cited in the references and used in accordance with their licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released implementation code and result tables are documented by the repository README, which describes the analysis and training entry points, external data dependencies, and MIT code license. The README explicitly states that raw rollout buffers and training logs are not included in the repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects; all data is generated by an LLM policy interacting with the synthetic ALFWorld simulator.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so no IRB review was applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The trained policy under study is itself an LLM (Qwen2.5-7B-Instruct fine-tuned with LoRA), and its use as the agent being optimised by GRPO is documented throughout §4 and §C. LLMs were also used as a writing/editing aid for the manuscript, which does not affect the methodology or experimental results.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.