Learn Where Outcomes Diverge: Efficient VLA RL via Probabilistic Chunk Masking

Vaidehi Bagaria*, **Nikshep Grampurohit***, **Pulkit Verma**†
Indian Institute of Technology Madras

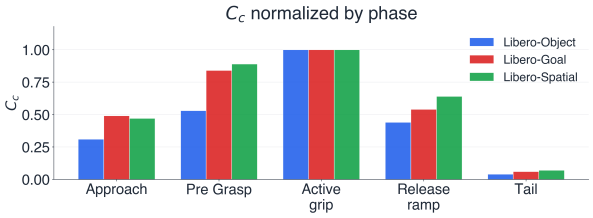
Abstract

Vision-language-action (VLA) policies are post-trained with reinforcement learning (RL) that uses binary task outcomes as world feedback, but this stage is computationally expensive. A natural response has been to speed up rollout data collection through faster simulators and world models. In GRPO-based VLA RL, however, we find that the dominant cost lies elsewhere: gradient computation accounts for $\sim$78% of wall-clock time per step in our runs, while rollout collection accounts for only $\sim$21%. Gradient cost dominates because much of this computation is spent on phases that contribute little to learning. GRPO's learning signal is driven by advantage variance: only phases where successful and failed rollouts diverge produce useful learning signal. However, GRPO assigns the same advantage to every chunk in a rollout. As a result, actor-update compute is spent uniformly across the trajectory, including phases the policy already handles after pre-training and supervised fine-tuning. This paper presents Probabilistic Chunk Masking (PCM), a drop-in modification to GRPO that allocates gradient computation to a small, probabilistically selected subset of chunks per trajectory. PCM scores semantic phases using success-failure action variance, a rollout-derived proxy for per-phase gradient variance, and samples a fixed chunk budget with online-updated phase-level keep probabilities. We formalize per-phase gradient variance as the quantity that determines where gradient computation is useful and show that success-failure action variance provides a measurable proxy for it. PCM requires no reward model or learned critic, exploiting the structure of binary world-feedback signals directly.. On three LIBERO benchmarks, PCM matches the final success rate of standard GRPO while achieving $2.38\times$ wall-clock speedup, $4.8\times$ faster gradient updates, and 60% lower peak activation memory, while backpropagating through fewer than 20% of trajectory chunks.

1 Introduction

Reinforcement learning (RL) has become essential for vision-language-action (VLA) models, enabling generalization to new environments and behaviors [Black et al., 2024, Kim et al., 2024]. The field has increasingly converged on Group Relative Policy Optimization (GRPO) [Shao et al., 2024] whose critic-free formulation removes the cost and complexity of training a separate value model [Li et al., 2025a, Lu et al., 2025]. As these systems scale, speeding up post-training has become critical. Efforts to speed up GRPO-based VLA RL have implicitly assumed that rollout collection is the bottleneck. We measure where time actually goes and find the reverse: in our runs gradient computation accounts for $\sim$78% of wall-clock time per training step, while rollout collection accounts for only $\sim$21% (Fig. 1, right).

*Equal contribution.
†Correspondence to pulkitv@cse.iitm.ac.in.



Benchmark	Grad (%)	Rollout (%)
LIBERO-Object	77.91	21.27
LIBERO-Goal	77.86	21.29
LIBERO-Spatial	77.85	21.29

Figure 1: Left: per-phase success–failure action variance (C_c) concentrates on decision-critical phases. Right: gradient computation dominates wall-clock training cost in simulator-based GRPO with 10 rollouts per prompt. Remaining $\sim$1% is logging, evaluation, and inter-step overhead.

This inefficiency arises from a deeper structural feature shared by most RL algorithms. Backpropagation runs over every chunk in the trajectory, and trajectory-level credit assigns the same advantage to each chunk regardless of which determined the outcome. Both inefficiencies share a common cause: the trajectory has been treated as the unit of gradient computation. Even methods that differentiate per-step credit, such as PPO [Schulman et al., 2017] with a learned critic, still apply forward/backward-pass compute to all sampled timesteps. They modify the signal, not the compute allocation.

This observation suggests a missing axis in VLA RL optimization: *within a trajectory, which chunks should receive gradient computation at all?*

The answer is not all of them. GRPO's learning signal is driven by differences in advantage across rollouts: when rollout rewards are uniform, gradients vanish [Wang et al., 2026, Xu and Ding, 2026]. We argue that variance non-uniformity persists even within a single non-collapsed group. Supervised fine-tuning is effective in densely covered, unimodal regions, but weak under sparse demonstrations or multimodal optimal behavior [Chu et al., 2025]; these are precisely the regimes that RL fine-tuning is intended to refine. This suggests that (a) gradient computation should be treated as an allocatable resource and (b) recomputing gradients on phases the base policy already handles wastes compute and memory. From this perspective, RL training becomes a problem of allocating limited gradient budget to the parts of trajectories that carry useful learning signal.

We present Probabilistic Chunk Masking (PCM), a drop-in modification to GRPO that updates only a fixed budget of chunks per trajectory. PCM assigns phase-level keep probabilities using the success-failure action variance C_c, which is concentrated in a small number of outcome-critical semantic phases c (Fig. 1, left). It then physically removes the unselected chunks from the backward pass, significantly reducing actor-update compute and activation memory. We further prove that C_c serves as a measurable lower-bound proxy for the true per-phase gradient variance V_c, giving a principled basis for this allocation. The **main contributions** of this paper are as follows:

- **We show that V_c is the right signal for gradient allocation** and derive a Neyman allocation proportional to $N_c\sqrt{V_c}$ (N_c is the number of chunks in phase c), which concentrates gradient compute on outcome-divergent phases.
- **We establish that the per-phase success–failure action variance C_c is a sufficient proxy for V_c**, computable directly from rollouts GRPO already produces with no reward model or learned critic.
- **Empirically, these enable substantial efficiency gains in gradient-bound VLA-RL.** On LIBERO [Liu et al., 2023], PCM achieves 2.38$\times$ wall-clock speedup to 98% convergence, 4.8$\times$ faster gradient steps, 60% lower peak activation memory, using less than 20% of chunks per trajectory.

2 Related works

Reinforcement Learning Fine-Tuning of VLA Models. Reinforcement learning has become an effective post-training stage for vision-language-action (VLA) models [Chu et al., 2025, Liu et al., 2025]. Early approaches relied on learned critics [Hu et al., 2024, Guo et al., 2025, Chen et al., 2025a, Tan et al., 2025], but the field has converged on critic-free GRPO-style optimization as these systems have scaled [Li et al., 2025a, Lu et al., 2025, Chen et al., 2025b, Ye et al., 2025].

To support this shift, advances in world models and simulation have significantly reduced rollout cost [Zhu et al., 2025, Liu et al., 2026, Zang et al., 2026] under the implicit assumption that data collection

is the bottleneck. However, this progress has left gradient computation, which we observe accounts for $\sim$78% of training time, largely unoptimized. Other work improves the source or structure of the reward signal [Li et al., 2025b, Zhai et al., 2025, Tan et al., 2025] by altering the reward, critic, rollout source, or advantage/credit-assignment rule. We show that this additional machinery can be avoided: C_c (success-failure action variance within a phase), an internal signal already present in GRPO rollouts, directly captures where learning signal concentrates and is sufficient to guide gradient allocation.

Efficiency Methods for GRPO. GRPO's learning signal is driven by advantage variance [Nan et al., 2025], and prior large language model (LLM) RL work has exploited this at two granularities. At the prompt level, methods filter or reweight zero-variance rollout groups [Yu et al., 2025, Zheng et al., 2025, Hu et al., 2025]. At the token level, work selects which sequence positions receive gradient using policy-internal signals such as token entropy or probability shifts [Wang et al., 2025, Khandoga et al., 2026, Hu et al., 2026]. PCM operates at a third granularity: phase-level allocation within a trajectory. Entropy and probability shifts are indirect proxies that measure where the model is uncertain, not where successful and failed rollouts actually diverge. In settings with verifiable outcomes (like VLA RL), this raises a more fundamental question, *why estimate outcome-criticality indirectly when the outcome signal is already available?* We use C_c, computed directly from rollout outcomes, to allocate gradient compute according to observed success–failure divergence. Unlike prior work that relies on proxy signals such as entropy or probability shifts, PCM exploits outcome-grounded divergence.

Parameter-efficient methods like LoRA and its extensions are standard in LLM and VLA fine-tuning, reducing trainable parameters but not the activation cost of backpropagating through long trajectories [Hu et al., 2021, Aghajanyan et al., 2021, Dettmers et al., 2023, Zhang et al., 2025, 2023]. Activation-focused methods reduce this memory cost through checkpointing and streaming backpropagation [Chen et al., 2016, Korthikanti et al., 2023, Luo et al., 2025]. Crucially, prior methods do not address the inefficiency of computing gradients on outcome-invariant phases within a trajectory, which still incur computation cost despite contributing little to learning.

3 Preliminaries

We briefly describe the setup for chunk-level GRPO, which our method builds on.

VLA Policies and Action Chunks. Let π_θ denote a VLA policy that produces actions in chunks of length L conditioned on observation $s_{i,k}$, where i indexes the trajectory and k the chunk within that trajectory. A rollout of trajectory i decomposes into $N_i = \lceil T_i/L \rceil$ chunks, where T_i is the total number of timesteps.

Group Relative Policy Optimization. GRPO [Shao et al., 2024] samples a group of G rollouts from the current policy, scores each with a binary reward $r_i \in \{0, 1\}$, and forms the group-relative advantage

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

Writing $a_{i,k}$ for the action tokens of chunk k in trajectory i, the chunk-level GRPO objective is

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_i\left[\sum_{k=1}^{N_i} A_i \cdot \log \pi_\theta(a_{i,k} \mid s_{i,k})\right], \tag{2}$$

with PPO-style clipping omitted for clarity.

4 Methodology: Probabilistic Chunk Masking

4.1 Phase-Conditioned Gradient Variance

Following the VLA-RL setup formalized in Appendix A, we partition every trajectory into K semantic phases using a deterministic labeling rule (described in Sec. 5.1). Let $\mathcal{P} = \{c_1, \ldots, c_K\}$ denote this phase set, and let $\varphi(i, k) \in \mathcal{P}$ denote the phase of chunk (i, k).

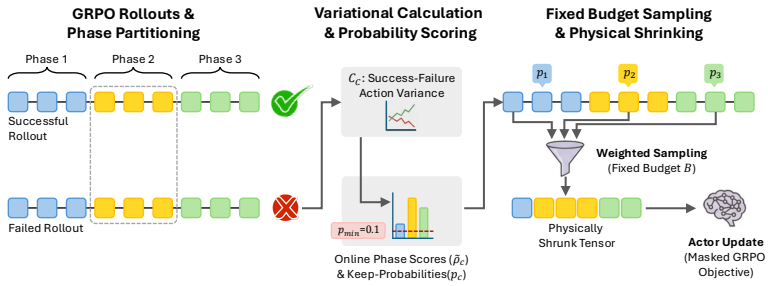


Figure 2: Probabilistic Chunk Masking (PCM). We compute per-phase success–failure action variance (C_c) from GRPO rollouts, use it to prioritize decision-critical phases, and sample a fixed budget of chunks per trajectory. Gradients are computed only on selected chunks.

Phase decomposition of the GRPO gradient. The gradient of $\mathcal{L}_{\text{GRPO}}$ decomposes by phase:

$$\nabla_\theta \mathcal{L}_{\text{GRPO}} = \sum_{c \in \mathcal{P}} g_c(\theta), \qquad g_c(\theta) = -\mathbb{E}_i \left[A_i \sum_{k\,:\,\varphi(i,k)=c} \nabla_\theta \log \pi_\theta(a_{i,k} \mid s_{i,k}) \right]. \tag{3}$$

The learning signal available in phase c is characterized by the *per-phase gradient variance*

$$V_c = \text{Var}(A_i \cdot \nabla_\theta \log \pi_\theta(a_{i,k} \mid s_{i,k}) \mid \varphi(i,k) = c)\,. \tag{4}$$

Intuitively, V_c is large for phases where the policy behaves differently across successful and failed rollouts. Throughout, V_c refers to the within-trajectory chunk variance (A_i fixed); under advantage normalization (Eq. 1), this matches the unconditional variance in Eq. (4) up to an $O(1)$ constant. We formalize this observation as follows. Note that formal proofs appear in Appendix B.

Lemma 1 (Phase gradient variance)**.** *Let $c \in \mathcal{P}$ be a phase in which π_θ has converged, i.e., the action distributions of successful and failed rollouts are identical in phase c. Then $\|g_c(\theta)\| \approx 0$ and $V_c \approx 0$, so gradient samples from phase c contribute negligible signal. Conversely, for a phase c in which the base policy is underspecified, V_c is large and gradient samples from phase c are informative.*

Empirical proxy for $\mathbf{V_c}$. Computing V_c exactly requires access to the gradient distribution, which is expensive. We instead use the *success-failure action variance*

$$C_c = \left\| \mathbb{E}[a_{i,k} \mid r_i = 1,\, \varphi(i,k) = c] - \mathbb{E}[a_{i,k} \mid r_i = 0,\, \varphi(i,k) = c] \right\|, \tag{5}$$

computable directly from the rollout group that GRPO already produces, with no auxiliary model or annotation. We compute C_c only for rollout groups with non-zero reward variance.

Lemma 2 (C_c as a proxy for V_c)**.** *For a policy π_θ that is locally Gaussian with per-dimension variance σ_π^2, the per-phase gradient variance satisfies $V_c \geq C_c^2 / 4\sigma_\pi^2$.*

Note that the bound in Lemma 2 is a lower bound that absorbs $\|\nabla_\theta \mu_\theta(s)\|^2$ into the constant. It justifies using C_c to recover the *ordering* of V_c across phases, rather than exact correspondence. We treat C_c as a practical proxy supported by empirical evidence: the realized allocation (Fig. 6) closely matches the $\sqrt{V_c}$-weighted prediction, and variance-aware ablations (Fig. 5b) show that C_c-weighted selection outperforms both random masking and highest-variance-only selection.

4.2 Optimal Budget Allocation

We now state the core theoretical result. Given a fixed chunk budget B per trajectory and phase variance estimates $\{V_c\}_{c \in \mathcal{P}}$, we ask: how should B be distributed across phases to minimize the variance of the GRPO gradient estimator (the sampling noise around the true gradient, distinct from the per-phase signal V_c)? Lower estimator variance means a more accurate estimate of the true gradient per step, which translates directly to faster stochastic gradient descent (SGD) convergence [Bottou et al., 2018]. This follows Neyman-allocation intuition: under a fixed sampling budget, phases should be sampled in proportion to their contribution to estimator variance.

Theorem 1 (Optimal phase allocation)**.** *Let b_c denote the number of chunks sampled from phase c, with $\sum_{c \in \mathcal{P}} b_c = B$, and let N_c denote the expected number of chunks in phase c per trajectory. The*

allocation minimizing the variance of the unbiased GRPO gradient estimator (Eq. (2) subject to the budget constraint is

$$b_c^* = B \cdot \frac{N_c\sqrt{V_c}}{\sum_{c' \in \mathcal{P}} N_{c'}\sqrt{V_{c'}}}. \tag{6}$$

The speedup over uniform allocation depends on how sharply V_c concentrates across phases. When V_c is uniform across all K phases, Eq. (6) reduces to uniform allocation $b_c = B/K$. When V_c concentrates on a small subset of phases, the Neyman allocation assigns disproportionately more budget to those outcome divergent phases, reducing estimator variance relative to uniform at the same total budget B. The concentration of C_c on contact-rich phases observed in Fig. 1, combined with Lemma 2, implies that V_c is similarly concentrated, making this regime directly applicable here. Section 5 quantifies the resulting wall-clock speedup empirically. A formal convergence-rate derivation and speedup formula are provided in Appendix B.1.

4.3 Online Phase Score Estimation

Theorem 1 requires knowledge of $\{V_c\}$, which is not directly observable. By Lemma 2, the relative ordering of $\{V_c\}$ is captured by the rollout-computable proxy $\{C_c\}$. We therefore use C_c to set per-phase weights; since phase c contains N_c candidate chunks, weighted sampling consistent with Eq. (6) gives $\mathbb{E}[b_c] \propto N_c\sqrt{V_c}$, concentrating gradient computation on outcome-divergent phases. Phases with larger estimated V_c receive higher keep probability, while larger N_c contributes more candidate chunks. Both factors increase sampling frequency within the fixed budget B.

During training, we compute a batch-level phase score $C_c^{(t)}$ from the rollout group at step t. We maintain a short buffer $\mathcal{B}_c$ for each phase and append the current score to the short refresh buffer. With $G = 10$ rollouts and binary rewards, individual C_c estimates may be noisy, especially under imbalanced success-failure splits. Therefore, scores are averaged over $T_{\text{rc}} = 5$ batches before refreshing the keep probabilities.

$$\mathcal{B}_c \leftarrow \mathcal{B}_c \cup \{C_c^{(t)}\}. \tag{7}$$

Every T_{rc} steps (we use $T_{\text{rc}} = 5$), we collapse the buffered scores into a share-normalized phase score:

$$S_c = \sum_{C_c^{(t)} \in \mathcal{B}_c} C_c^{(t)}, \qquad \rho_c = \frac{S_c}{\sum_{c'} S_{c'}}, \qquad \tilde{\rho}_c = \frac{\rho_c}{\max_{c'} \rho_{c'}} \in [0, 1]. \tag{8}$$

Share normalization is scale-invariant: it preserves the relative phase ordering even though the absolute magnitude of C_c can vary across tasks and training stages. The resulting $\tilde{\rho}_c$ is used as the phase-level score for probabilistic chunk selection. For the first update, the keep probabilities are computed from the phase scores of the first rollout group itself; after that, scores are updated every T_{rc} steps using the buffered window. This online update lets PCM track the realized learning signal: if a phase learns faster, its success–failure divergence decreases and keep probability naturally falls.

4.4 Probabilistic Mask Selection

Given phase scores $\{\tilde{\rho}_c\}$ as proxies for $\{\sqrt{V_c}\}$, we map them to per-phase keep-probabilities that implement the allocation of Theorem 1 in expectation:

$$p_c = \max(p_{\min},\ \tilde{\rho}_c) \in [p_{\min}, 1], \tag{9}$$

with $p_{\min} = 0.1$. The floor $p_{\min}$ prevents any phase from being structurally excluded between buffer recomputes. This is important because $\tilde{\rho}_c$ may be transiently zero for a phase with nonzero true variance V_c if that phase is underrepresented or temporarily underestimated in the current buffer window. Excluding such phases entirely would violate the exploration requirement of the allocation and could cause the estimated $\{\rho_c\}$ to diverge from the true $\{V_c\}$ ordering over time. Each chunk inherits its phase's probability weight: $w_{i,k} = p_{\varphi(i,k)}$. The set $\{p_c\}$ is held fixed across the next T_{rc} steps.

4.5 Fixed-Budget Sampling and Physical Shrinking

Given the allocation rule derived in Theorem 1, we sample a fixed budget of B chunks per trajectory using weighted sampling without replacement [Plackett, 1975, Luce, 1959], with weights $\{w_{i,k}\}$:

$$\mathcal{K}_i \sim \text{WeightedSample}(w_{i,1}, \ldots, w_{i,N_i};\ \min(B, N_i),\ \text{w/o replacement}). \tag{10}$$

The selected set is $\mathcal{K}_i = \{k_1, \ldots, k_{m_i}\}$. Non-selected chunks are physically removed from the batch tensor before the forward pass.

The actor update uses the masked objective:

$$\mathcal{L}_{\text{PCM}}(\theta) = -\mathbb{E}_i \left[\sum_{k \in \mathcal{K}_i} A_i \cdot \log \pi_\theta(a_{i,k} \mid s_{i,k}) \right] \tag{11}$$

The masked objective in Eq. (11) omits the $1/p_c$ importance weights, making it a biased estimator of the full-trajectory GRPO gradient. The bias is $\sum_{c \in \mathcal{P}} (1 - p_c)\, g_c$: chunks that are not selected contribute zero to Eq. (11) but g_c to the full gradient. Under our allocation $p_c \propto \sqrt{V_c}$, the factor $(1 - p_c)$ is large precisely when V_c is small, and by Lemma 1, small V_c implies small $\|g_c\|$. Each term in the bias is therefore suppressed by at least one small factor, so Theorem 1's allocation remains approximately optimal under the biased estimator. The trade-off is favorable: at the same chunk budget B, the masked estimator has substantially lower variance than the importance-weighted unbiased alternative, translating to faster SGD convergence (Appendix B.1). A formal bound is given in Appendix B.2.

Algorithm 1 Probabilistic Chunk Masking for GRPO

Require: Policy π_θ; group size G; chunk budget B; floor $p_{\min}$; refresh window T_{rc}

Initialize phase-score buffers $\mathcal{B}_c \leftarrow [\,]$ for all phases c
while not converged **do**
  Sample G rollouts and compute binary rewards $\{r_i\}$ ▷ Eq. (1)
  Compute GRPO advantages $\{A_i\}$ from the group rewards ▷ Eq. (1)
  Label each chunk with a phase $\varphi(i, k)$ from the gripper trajectory ▷ Sec. 5.1
  Compute per-phase success-failure action variance $\{C_c^{(t)}\}_{c \in \mathcal{P}}$ from rollouts ▷ Eq. (5)
  Append $C_c^{(t)}$ to buffer $\mathcal{B}_c$ for all $c \in \mathcal{P}$ ▷ Eq. (7)
  if first batch **or** $|\mathcal{B}_c| = T_{\text{rc}}$ **then**
    Collapse buffers, share-normalize and max-normalize ▷ Eq. (8)
    Apply floor: $p_c \leftarrow \max(p_{\min}, \tilde{\rho}_c)$ for all c ▷ Eq. (9)
    Reset phase buffers: $\mathcal{B}_c \leftarrow [\,]$ for all $c \in \mathcal{P}$
  end if
  for each trajectory i with N_i chunks **do**
    Set chunk weights $w_{i,k} \leftarrow p_{\varphi(i,k)}$ for $k = 1, \ldots, N_i$ ▷ Sec. (4.4)
    Sample $\mathcal{K}_i$ with $|\mathcal{K}_i| = \min(B, N_i)$ via weighted sampling w/o replacement ▷ Eq. (10)
    Physically remove non-selected chunks from batch; keep only $\mathcal{K}_i$ for actor update ▷ Sec. 4.5
  end for
  Update π_θ using the masked GRPO loss ▷ Eq. (11)
end while

5 Empirical Evaluation

5.1 Experimental Setup

Models and Benchmarks. We run our experiments on OpenVLA-OFT [Kim et al., 2025], a 7B vision-language-action model that predicts 7-DoF action chunks with chunk length L=8. We evaluate on three benchmarks: LIBERO-Object, LIBERO-Spatial, and LIBERO-Goal [Liu et al., 2023].

Phase Labeling. We assign each chunk to one of K=5 semantic phases using a deterministic, multi-grasp-aware rule over the per-chunk gripper-close fraction $g_f[j] \in [0, 1]$. Active-grip chunks ($g_f[j] \geq 0.5$) capture sustained grasp and transport; pre-grasp labels the up-to-three chunks immediately before sustained closure ($0.1 \leq g_f[j] < 0.5$); release-ramp labels the up-to-three chunks after; approach covers the remaining pre-contact chunks; and tail covers post-release open-gripper chunks; Overlapping windows are resolved by the priority ordering `active-grip` > `pre-grasp` > `release-ramp` > `approach` > `tail`. This rule is applied identically to successful and failed trajectories, enabling C_c to compare outcomes under a shared phase partition. Full details are in Appendix E.2.

Training and Evaluation. We implement PCM on the SimpleVLA-RL [Li et al., 2025a] verl [Sheng et al., 2025] pipeline with GRPO groups of 10 rollouts per prompt, comparing against the full-trajectory chunk-level GRPO baseline. We fine-tune OpenVLA-OFT with LoRA using 2 NVIDIA H100 GPUs. We evaluate every training step using 50 held-out validation rollouts. Results are averaged across three seeds, and we report per-step update time and peak GPU memory. Detailed experimental settings are in Appendix E.

Research Questions. Our experiments are designed to answer the following:

- **RQ1** Does variance-aware chunk selection preserve the learning dynamics of full-trajectory GRPO, measured by step-wise accuracy curves?
- **RQ2** Does selective gradient allocation reduce wall-clock time to reach a target success rate without affecting sample efficiency?
- **RQ3** How sensitive is performance to the chunk budget B, and is there a stable operating point that balances gradient signal with per-step speedup?
- **RQ4** Is variance-aware probabilistic selection necessary, or do simpler masking strategies achieve comparable performance under the same chunk budget?

5.2 Results and Discussion

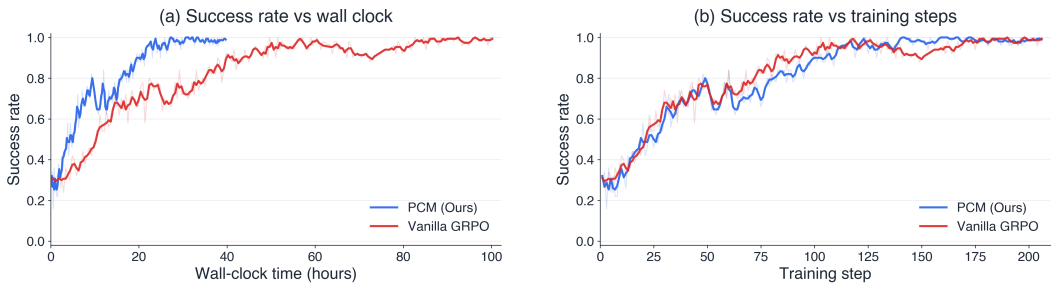


Figure 3: Success rate as a function of wall-clock time **(a)** and training step **(b)** on LIBERO-Object, averaged over three seeds. PCM (B=12) closely matches full-trajectory GRPO across training steps while converging significantly faster.

RQ1: Learning Dynamics. We compare PCM with chunk budget B=12 against full-trajectory GRPO on LIBERO-Object using the same SFT-initialized OpenVLA-OFT checkpoint, identical rollout groups, rewards, advantages, and optimizer settings. The methods differ only in whether the actor update is computed over all chunks or the subset selected by PCM. Both are trained for a fixed budget of 200 steps.

As a function of training step (Fig. 3b), PCM and GRPO exhibit nearly identical learning curves and reach matched final accuracy. This shows that PCM preserves per-step learning dynamics while improving efficiency, confirming that *the selective gradient allocation does not degrade step-wise convergence behavior*.

RQ2: Wall-Clock Efficiency. As shown in Fig. 3a, PCM reaches near-saturation significantly earlier in wall-clock time. Fig. 4a shows that PCM reaches the target success rate of 98% up to 2.38× faster than vanilla GRPO, with the gap widening sharply at higher accuracy thresholds. Table 1 shows that this trend holds across all LIBERO benchmarks, reducing the average time to 98% success from 48.97 to 20.55 hours. To understand how this speedup decomposes, Fig. 4b-c separate per-step compute from overall convergence. PCM reduces activation memory by 60% (10.1 → 4.1 GB) and peak GPU memory by 15% (39.7 → 33.6 GB) per step.

	Vanilla GRPO	PCM (Ours)
LIBERO-Object	45.78	**19.23**
LIBERO-Goal	51.25	**21.18**
LIBERO-Spatial	49.89	**21.23**
Overall	48.97	**20.55**

Table 1: Wall-clock time (hours) to reach 98% success rate. Lower is better.

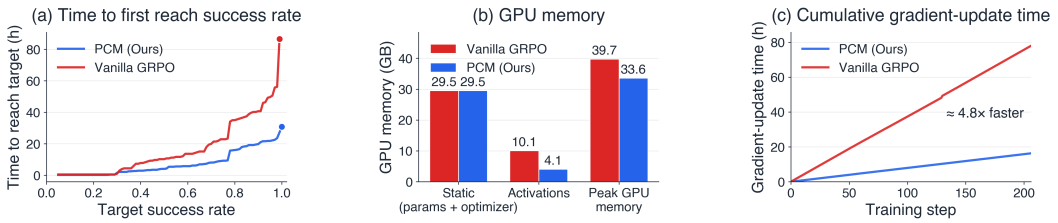


Figure 4: Efficiency on LIBERO-Object at B=12. **(a)** Wall-clock time to first reach each SR threshold; the PCM/GRPO gap widens at higher thresholds. **(b)** Activation memory is reduced by 60%. **(c)** Cumulative actor-update time over 200 steps is $4.8\times$ faster.

Cumulative gradient-update time over 200 training steps is $4.8\times$ faster. Because the per-step learning curves are matched (Fig. 3b), the wall-clock gain is entirely attributable to per-step compute savings rather than improved sample efficiency. Together these results show that the *selective gradient allocation substantially reduces per-step cost while leaving sample efficiency intact*.

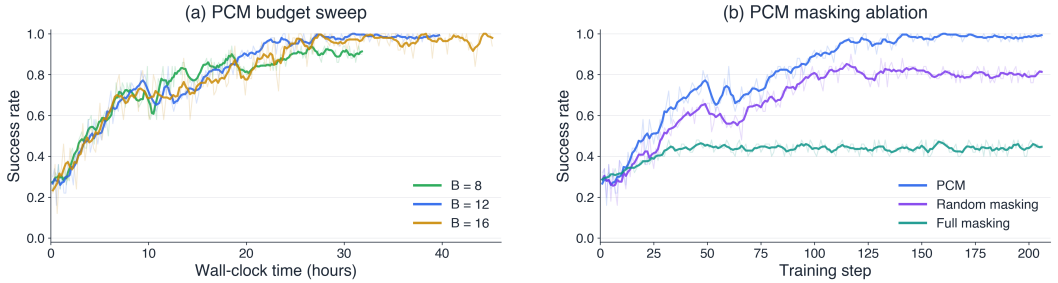


Figure 5: Ablations on LIBERO-Object. **(a)** Chunk budget sweep shows that B=12 preserves final success while retaining most of the wall-clock speedup. **(b)** At fixed B=12, PCM outperforms random masking and highest-variance-phase selection.

RQ3: Sensitivity to Chunk Budget B. PCM samples a fixed budget of B chunks per trajectory. Smaller budgets improve per-step efficiency but reduce gradient coverage, while larger budgets better approximate full GRPO at higher cost. We sweep B $\in$ {8, 12, 16} on LIBERO-Object using the same training configuration. We select this range for B to bracket the inflection of the cumulative C_c curve in Fig. 8, plotted against the fraction of trajectory chunks retained, which marks the smallest budget capturing most of the available learning signal. Details in Appendix D.

Fig. 5a shows that all budgets reach comparable final accuracy, but differ in efficiency. B=8 is fastest per step but has degraded sample efficiency and plateaus at a slightly lower accuracy, indicating insufficient gradient signal. B=16 provides no accuracy gain over B=12 while increasing convergence time, showing diminishing returns beyond high-C_c phases. B=12 lies at the inflection point, matching final accuracy while preserving the wall-clock advantage. We use B=12 as our default for all other experiments. B=12 is large enough to consistently sample from high-C_c phases under the probabilistic selection rule while remaining small enough to deliver substantial per-step compute and memory savings.

RQ4: Variance Concentration and the Role of Variance-Aware Selection. To test whether variance-aware probabilistic selection is necessary, we compare PCM at B=12 against two ablations that occupy opposite extremes of the allocation spectrum: *random masking*, which samples B chunks uniformly at random from each trajectory (no concentration on high-C_c phases), and *full masking*, which restricts updates to the single highest-C_c phase and discards all other chunks (maximum concentration, no exploration). Together, these bracket PCM, which lies between these extremes via probabilistic C_c-weighted sampling.

Fig. 5b shows that both ablations underperform PCM. Random masking plateaus at $\sim$78% success rate, 22 points below PCM at the same chunk budget. Full masking fails more sharply, plateauing

at a low accuracy of $\sim$43%. Since all methods use the same B=12 budget, these gaps isolate the contribution of variance-aware selection from compute reduction.

Fig. 6 shows PCM's realized gradient allocation across training, revealing three mechanisms that together explain its performance.

- **Concentration:** Active-grip ($\sim$7–8 chunks per trajectory) and pre-grasp ($\sim$3–5 chunks per trajectory) dominate throughout, matching the high-C_c phases (Fig. 1) and the $\sqrt{V_c}$-weighted allocation predicted by Theorem 1. The agreement between predicted and realized allocation is the empirical confirmation that C_c tracks V_c in practice, validating Lemma 2's proxy assumption.

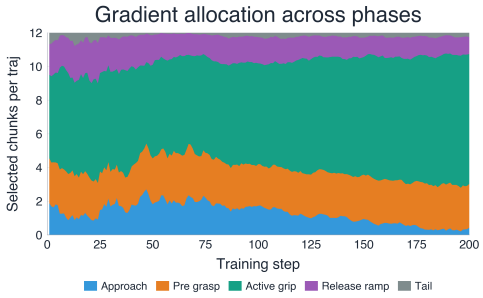


Figure 6: Phase-wise gradient allocation under PCM over training steps.

- **Exploration:** Probabilistic C_c-weighted sampling (Eq. 10) ensures that every phase has nonzero selection probability whenever its weight is above zero, providing the primary exploration mechanism. The $p_{\min} = 0.1$ floor (Eq. 9) acts as a safety net for low-scoring phases (e.g., *tail*): it prevents transient zero estimates from collapsing a phase's weight to zero between buffer recomputes, which would otherwise cause online phase scores to fail to recover when the true V_c ordering shifts during training.

- **Adaptation:** Approach and release-ramp allocations decline from $\sim$2 to $\sim$1 chunks and from $\sim$2.5 to $\sim$1.5 chunks, respectively, as the policy masters those phases, with budget shifting toward phases where learning signal concentrates. This online dynamic that adapts with evolving success-failure gap of the policy cannot be reproduced by static weighting.

The ablations occupy degenerate corners of this design space. Random masking explores but lacks concentration, matching the suboptimal uniform-allocation regime in Theorem 1, where $b_c = B/K$ (equal budget across phases) is inefficient when V_c is non-uniform across phases. Full masking concentrates updates on the highest-C_c phase but discards weaker adjacent-phase signals and prevents exploration and adaptation.

Only PCM has all three properties: C_c-weighted keep probabilities with a $p_{\min}$ floor to concentrate updates on contact-rich phases while retaining lower-scoring phases. The ablations show that variance-aware probabilistic selection substantially outperforms uninformed masking strategies in exploiting the concentrated structure of C_c.

6 Limitations and Future Work

Our evaluation spans the three LIBERO suites (Object, Spatial, Goal), which test transfer of object, spatial, and task knowledge respectively (Table 1). We do not test longer horizons or bimanual coordination; however, the underlying allocation principle applies whenever V_c is non-uniform, a structural feature of any GRPO setting where the policy has not converged uniformly. Experiments use a gripper-based phase partition rule. Extension to other settings is straightforward in principle (any consistent decomposition suffices), though we do not validate alternative phase abstractions empirically. We do not directly benchmark against methods targeting sample efficiency through different objectives (e.g., entropy-based token selection or prompt filtering). These methods differ from PCM in both granularity (token vs. phase) and signal (policy-internal uncertainty vs. outcome-grounded divergence); adapting them to the VLA chunk-level setting is beyond the scope of this work.

Future work includes extending PCM to longer-horizon and bimanual tasks, validating learned temporal phase segmenters or LLM-derived phase labelers, and applying the variance-allocation principle to LLM reasoning where C_c can be computed against verified outcomes. Combining PCM with orthogonal sample-efficiency methods remains an open direction.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2021. doi: 10.18653/v1/2021.acl-long.568. URL https://arxiv.org/abs/2012.13255v1.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control, 2024. URL https://arxiv.org/abs/2410.24164v4.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. abs/1604.06174, 2016. URL http://arxiv.org/abs/1604.06174.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. ConRFT: A Reinforced Fine-tuning Method for VLA Models via Consistency Policy, 2025a. URL https://arxiv.org/abs/2502.05450v2.

Zengjue Chen, Runliang Niu, He Kong, Qi Wang, Qianli Xing, and Zipei Fan. TGRPO :Fine-tuning Vision-Language-Action Model via Trajectory-wise Group Relative Policy Optimization, 2025b. URL https://arxiv.org/abs/2506.08440.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. In *International Conference on Machine Learning*, 2025. doi: 10.48550/arxiv.2501.17161. URL https://arxiv.org/abs/2501.17161v2.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023. doi: 10.48550/arxiv.2305.14314. URL https://arxiv.org/abs/2305.14314v1.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving Vision-Language-Action Model with Online Reinforcement Learning. In *IEEE International Conference on Robotics and Automation*, 2025. URL https://arxiv.org/abs/2501.16664v1.

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*, 2021. URL https://arxiv.org/abs/2106.09685v2.

Jiaheng Hu, Rose Hendrix, Ali Farhadi, Aniruddha Kembhavi, Roberto Martin-Martin, Peter Stone, Kuo-Hao Zeng, and Kiana Ehsani. FLaRe: Achieving Masterful and Adaptive Robot Policies with Large-Scale Reinforcement Learning Fine-Tuning. In *IEEE International Conference on Robotics and Automation*, 2024. doi: 10.1109/icra55743.2025.11127934. URL https://arxiv.org/abs/2409.16578v2.

Yuelin Hu, Zhengxue Cheng, Wei Liu, and Li Song. Entropy-Gated Selective Policy Optimization:Token-Level Gradient Allocation for Hybrid Training of Large Language Models, 2026. URL https://arxiv.org/abs/2602.03309.

Zengjie Hu, Jiantao Qiu, Tianyi Bai, Haojin Yang, Binhang Yuan, Qi Jing, Conghui He, and Wentao Zhang. VADE: Variance-Aware Dynamic Sampling via Online Sample-Level Difficulty Estimation for Multimodal RL, 2025. URL https://arxiv.org/abs/2511.18902.

Mykola Khandoga, Rui Yuan, and Vinay Kumar Sankarapu. Beyond Uniform Credit: Causal Credit Assignment for Policy Optimization, 2026. URL https://arxiv.org/abs/2602.09331.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *Conference on Robot Learning*, 2024. doi: 10.48550/arxiv.2406.09246. URL https://arxiv.org/abs/2406.09246v3.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Robotics: Science and Systems*, 2025. URL https://arxiv.org/abs/2502.19645.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing Activation Recomputation in Large Transformer Models. In *Proceedings of the Machine Learning and Systems Conference (MLSys)*, 2023. URL https://openreview.net/forum?id=nvRncppDoD.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA Training via Reinforcement Learning, 2025a. URL https://arxiv.org/abs/2509.09674v1.

Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. VLA-RFT: Vision-Language-Action Reinforcement Fine-tuning with Verified Rewards in World Simulators, 2025b. URL https://arxiv.org/abs/2510.00406.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.03310.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What Can RL Bring to VLA Generalization? An Empirical Study, 2025. URL https://arxiv.org/abs/2505.19789v4.

Xiaokang Liu, Zechen Bai, Hai Ci, Kevin Yuchen Ma, and Mike Zheng Shou. World-VLA-Loop: Closed-Loop Learning of Video World Model and VLA Policy, 2026. URL https://arxiv.org/abs/2602.06508.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards Masterful and General Robotic Manipulation with Scalable Reinforcement Learning, 2025. URL https://arxiv.org/abs/2505.18719v1.

R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, New York, 1959.

Qijun Luo, Mengqi Li, Lei Zhao, and Xiao Li. StreamBP: Memory-Efficient Exact Backpropagation for Long Sequence Training of LLMs. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.03077v1.

Gongrui Nan, Siye Chen, Jing Huang, Mengyu Lu, Dexun Wang, Chunmei Xie, Weiqi Xiong, Xianzhou Zeng, Qixuan Zhou, Yadong Li, and Xingzhong Xu. NGRPO: Negative-enhanced Group Relative Policy Optimization, 2025. URL https://arxiv.org/abs/2509.18851.

R. L. Plackett. The analysis of permutations. 24(2):193–202, 1975. doi: 10.2307/2346567.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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, 2024. URL https://arxiv.org/abs/2402.03300v3.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive Post-Training for Vision-Language-Action Models, 2025. URL https://arxiv.org/abs/2505.17016v1.

Hongcheng Wang, Yinuo Huang, Sukai Wang, Guanghui Ren, and Hao Dong. Why Tree-Style Branching Matters for Thought Advantage Estimation in GRPO, 2026. URL https://arxiv.org/abs/2509.24494.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 Rule: High-Entropy Minority Tokens Drive Effective Reinforcement Learning for LLM Reasoning. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.01939.

Zhongwen Xu and Zihan Ding. Single-stream Policy Optimization. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=b61UW62K7W.

Angen Ye, Zeyu Zhang, Boyuan Wang, Xiaofeng Wang, Dapeng Zhang, and Zheng Zhu. VLA-R1: Enhancing Reasoning in Vision-Language-Action Models, 2025. URL https://arxiv.org/abs/2510.01623.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2503.14476v2.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Peihong Wang, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. RLinf-VLA: A Unified and Efficient Framework for Reinforcement Learning of Vision-Language-Action Models, 2026. URL https://arxiv.org/abs/2510.06710.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A Vision-Language-Action-Critic Model for Robotic Real-World Reinforcement Learning, 2025. URL https://arxiv.org/abs/2509.15937.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Yang You, Guiming Xie, Xuejian Gong, and Kunlong Zhou. Train Small, Infer Large: Memory-Efficient LoRA Training for Large Language Models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.13533v2.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2303.10512.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act Only When It Pays: Efficient Reinforcement Learning for LLM Reasoning via Selective Rollouts. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=x5lITYXmW2.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. WMPO: World Model-based Policy Optimization for Vision-Language-Action Models, 2025. URL https://arxiv.org/abs/2511.09515v1.

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 state that PCM reallocates gradient computation based on intra-trajectory variance without altering the underlying GRPO objective. This is directly supported by both the theoretical formulation and empirical results, which show matched step-wise performance with reduced computation.

   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: Limitations are discussed in the main body of the paper.

   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: All theoretical proofs are complete and assumptions are clearly stated.

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: The paper specifies the training setup in detail.

   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: [Yes]

Justification: Required code is provided in the supplemental material and will be open sourced after review.

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: All hyperparameters along with reasons for choosing them are provided in the paper.

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: [No]

Justification: We evaluate using 3 inference seeds, with each evaluation step computed over 50 validation rollouts (10,000 total per training run). This matches the standard protocol in VLA-RL papers at this scale (e.g., SimpleVLA-RL [Li et al., 2025a]) and LLM RL (GRESO [Zheng et al., 2025]). Full confidence intervals on training-time measurements would require additional 7B-parameter training runs that exceed our compute budget.

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: [Yes]

   Justification: Compute requirements are mentioned in the main paper.

   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: The research complies with the NeurIPS Code of Ethics.

   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: [N/A]

    Justification: The paper optimizes existing research and would enable faster GRPO training. We do not foresee any fundamental negative societal impacts.

    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: The publicly available data and pretrained models used in this work do not pose a high risk of misuse, and we do not release any new models or datasets that would require additional safeguards.

    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: All licenses are adhered to and credit is given where due.

    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: [N/A]

    Justification: We do not release any new datasets or assets.

    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: We do not conduct any research with human subjects.

    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: We do not research with human subjects.

    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: [N/A]

    Justification: The core method development for our work does not involve LLMs.

    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.

A Problem Formulation and Assumptions

For clarity, we explicitly restate the problem formulation and assumptions underlying PCM. While these are implicit in the main text, we formalize them here to precisely define the input setting, the chunk-selection objective, and the structural conditions required for the analysis.

Input. A VLA policy π_θ^0 initialized from a supervised fine-tuning checkpoint, a simulated environment $\mathcal{E}$, a distribution over tasks $\mathcal{T}$ with binary reward $r : \tau \to \{0, 1\}$, and a group of G rollouts $\{\tau_i\}_{i=1}^G$ where each trajectory decomposes into $N_i = \lceil T_i/L \rceil$ chunks of length L, and a fixed gradient compute budget $B \ll \sum_i N_i$.

Output. A chunk selection policy $\mathcal{M} : \{\tau_i, r_i\}_{i=1}^G \to \{K_i \subseteq \{1, \ldots, N_i\}\}_{i=1}^G$ that minimizes

$$\sum_{i=1}^{G} \mathrm{Var}\left(\sum_{k \in K_i} A_i \cdot \nabla_\theta \log \pi_\theta(a_{i,k} \mid s_{i,k})\right) \tag{12}$$

subject to $|K_i| = B$ for all i (fixed compute budget), and $\mathcal{M}$ using only $\{\tau_i, r_i\}_{i=1}^G$ without access to auxiliary reward models or learned critics.

Assumptions. We assume the following:

- **Binary reward:** $r : \tau \to \{0, 1\}$ is trajectory-level, with each group containing both successes and failures ($0 < \mu_r < 1$); when all rollouts succeed or fail, advantages collapse and the variance signal is uninformative.
- **Fixed chunk length:** The policy produces actions in chunks of fixed length L, so the phase decomposition $\phi(i, k)$ is well-defined and consistent across trajectories.
- **Deterministic phase labeling:** $\phi(i, k) \in \mathcal{P}$ is computable from the gripper command trajectory alone, requiring no learned model.
- **Shared phase structure:** Successful and failed rollouts share the same phase partition $\mathcal{P}$, so per-phase action variance compares like with like across outcome groups.
- **SFT initialization:** π_θ^0 is initialized from a supervised fine-tuning checkpoint, ensuring gradient variance concentrates on a small subset of outcome-critical phases.

B Theoretical Results

In this section we provide proofs for the formal results in the main paper.

Lemma 1 (Phase gradient variance)**.** *Let $c \in \mathcal{P}$ be a phase in which π_θ has converged, i.e., the action distributions of successful and failed rollouts are identical in phase c. Then $\|g_c(\theta)\| \approx 0$ and $V_c \approx 0$, so gradient samples from phase c contribute negligible signal. Conversely, for a phase c in which the base policy is underspecified, V_c is large and gradient samples from phase c are informative.*

Proof. Since A_i is zero-mean by construction (Eq. (1)), we have $V_c = 0$ if and only if $\pi_\theta(a_{i,k} \mid s_{i,k})$ is identical across successful and failed rollouts for every chunk (i, k) with $\varphi(i, k) = c$. This holds exactly when outcomes are independent of the actions taken in phase c. When these conditional action distributions match, positive- and negative-advantage score-function terms are drawn from the same distribution and cancel in expectation, leaving $\|g_c(\theta)\| \approx 0$. When they diverge, phase c carries useful learning signal. □

Lemma 2 (C_c as a proxy for V_c)**.** *For a policy π_θ that is locally Gaussian with per-dimension variance σ_π^2, the per-phase gradient variance satisfies $V_c \geq C_c^2/4\sigma_\pi^2$.*

Proof. Let $\mu_c^+ = \mathbb{E}[a_{i,k} \mid r_i = 1, \varphi(i, k) = c]$ and $\mu_c^- = \mathbb{E}[a_{i,k} \mid r_i = 0, \varphi(i, k) = c]$. For a Gaussian policy $\pi_\theta(\cdot \mid s) = \mathcal{N}(\mu_\theta(s), \sigma_\pi^2 I)$, the score function is $\nabla_\theta \log \pi_\theta(a \mid s) = (a - \mu_\theta(s))/\sigma_\pi^2 \cdot \nabla_\theta \mu_\theta(s)$. Conditioning on the outcome group, the variance of the score is lower-bounded by the variance of the mean action across groups. Specifically, for any random variable X and a binary conditioning event E, $\mathrm{Var}(X) \geq \mathrm{Var}(\mathbb{E}[X \mid E])$ by the law of total variance. Applying this

with $X = A_i \cdot \nabla_\theta \log \pi_\theta(a_{i,k} \mid s_{i,k})$ and $E = \{r_i = 1\}$ versus $\{r_i = 0\}$, and using $|A_i| \le 1/\varepsilon$ as a finite bound, yields the equation in the lemma after absorbing the gradient norm $\|\nabla_\theta \mu_\theta\|^2$ and balanced-group constants into σ_π^2. The bound preserves the relative ordering of $\{V_c\}$ across phases. □

Theorem 1 (Optimal phase allocation)**.** *Let b_c denote the number of chunks sampled from phase c, with $\sum_{c\in\mathcal{P}} b_c = B$, and let N_c denote the expected number of chunks in phase c per trajectory. The allocation minimizing the variance of the unbiased ratio estimator of the GRPO gradient (Eq. (2)) subject to the budget constraint is*

$$b_c^* = B \cdot \frac{N_c\sqrt{V_c}}{\sum_{c'\in\mathcal{P}} N_{c'}\sqrt{V_{c'}}}. \tag{13}$$

Proof. We analyze the idealized stratified ratio estimator with deterministic per-phase budget bc; the masked loss in Eq. (11) implements this allocation in expectation under the sampling rule of Eq. (10), at the cost of a finite-sample bias bounded by the contribution of low-Vc phases (Sec. 4.5). The GRPO gradient decomposes by phase as in Eq. (3). PCM estimates the full phase gradient $g_c(\theta)$, which sums over all N_c chunks in phase c, via the ratio estimator

$$\hat{g}_c = \frac{N_c}{b_c} \sum_{k\in K_c} A_i \cdot \nabla_\theta \log \pi_\theta(a_{i,k} \mid s_{i,k}), \tag{14}$$

where $K_c \subseteq \{k : \varphi(i,k) = c\}$ with $|K_c| = b_c$ chunks drawn uniformly without replacement from the N_c available chunks in phase c. The N_c/b_c factor scales up the sampled sum to account for the full phase size, making $\hat{g}_c$ unbiased for $g_c(\theta)$:

$$\mathbb{E}_{K_c}[\hat{g}_c] = \frac{N_c}{b_c} \cdot b_c \cdot \frac{g_c(\theta)}{N_c} = g_c(\theta).$$

The variance of $\hat{g}_c$ is

$$\mathrm{Var}(\hat{g}_c) = \frac{N_c^2}{b_c^2} \cdot b_c \cdot V_c = \frac{N_c^2 V_c}{b_c}, \tag{15}$$

where V_c is the per-chunk variance within phase c (Eq. (4)), and the b_c in the numerator is the variance of a single-chunk sample, reduced by b_c draws.

The total estimator variance across all phases is

$$\sigma^2(\{b_c\}) = \sum_{c\in\mathcal{P}} \frac{N_c^2 V_c}{b_c}. \tag{16}$$

We minimize $\sigma^2(\{b_c\})$ subject to $\sum_{c\in\mathcal{P}} b_c = B$, $b_c \ge 0$. By the Cauchy-Schwarz inequality,

$$\left(\sum_{c\in\mathcal{P}} N_c\sqrt{V_c}\right)^2 = \left(\sum_{c\in\mathcal{P}} \frac{N_c\sqrt{V_c}}{\sqrt{b_c}} \cdot \sqrt{b_c}\right)^2 \le \underbrace{\left(\sum_{c\in\mathcal{P}} \frac{N_c^2 V_c}{b_c}\right)}_{\sigma^2(\{b_c\})} \cdot \underbrace{\left(\sum_{c\in\mathcal{P}} b_c\right)}_{B},$$

with equality if and only if $N_c\sqrt{V_c}/\sqrt{b_c} \propto \sqrt{b_c}$, i.e., $b_c \propto N_c\sqrt{V_c}$. Combining with $\sum_c b_c = B$ yields Eq. (13).

Substituting $b_c^* = B \cdot N_c\sqrt{V_c}/\sum_{c'} N_{c'}\sqrt{V_{c'}}$ into Eq. (16), the minimum total variance achieved is

$$\sigma_*^2 = \frac{1}{B}\left(\sum_{c\in\mathcal{P}} N_c\sqrt{V_c}\right)^2. \tag{17}$$

□

B.1 Convergence Derivation for Phase Allocation

We derive the convergence-rate comparison between the optimal allocation of Theorem 1 and uniform allocation. Under standard SGD assumptions (L-smooth objective, bounded per-phase gradient variance V_c), the number of steps to reach an ε-stationary point satisfies [Bottou et al., 2018]

$$T(\varepsilon) = \mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2}\right), \tag{18}$$

where σ^2 is the variance of the gradient estimator at sample budget B; the $1/B$ variance reduction from B-sample averaging is incorporated into σ^2 via the per-phase variance formulas.

Optimal allocation. From Theorem 1 and Eq. (17), the minimum achievable estimator variance under budget B is

$$\sigma_*^2 = \frac{1}{B}\left(\sum_{c\in\mathcal{P}} N_c\sqrt{V_c}\right)^2. \tag{19}$$

Substituting into Eq. (18):

$$T^*(\varepsilon) = \mathcal{O}\left(\frac{\left(\sum_{c\in\mathcal{P}} N_c\sqrt{V_c}\right)^2}{B\cdot\varepsilon^2}\right). \tag{20}$$

Uniform allocation. Under $b_c = B/K$ for all K phases, the total estimator variance is

$$\sigma^2_{\text{uniform}} = \sum_{c\in\mathcal{P}} \frac{N_c^2 V_c}{B/K} = \frac{K}{B}\sum_{c\in\mathcal{P}} N_c^2 V_c. \tag{21}$$

Substituting into Eq. (18):

$$T_{\text{uniform}}(\varepsilon) = \mathcal{O}\left(\frac{K\sum_{c\in\mathcal{P}} N_c^2 V_c}{B\cdot\varepsilon^2}\right). \tag{22}$$

Speedup ratio. Dividing Eq. (22) by Eq. (20):

$$\Delta = \frac{T_{\text{uniform}}(\varepsilon)}{T^*(\varepsilon)} = \frac{K\sum_{c\in\mathcal{P}} N_c^2 V_c}{\left(\sum_{c\in\mathcal{P}} N_c\sqrt{V_c}\right)^2}. \tag{23}$$

By the Cauchy-Schwarz inequality,

$$\left(\sum_{c\in\mathcal{P}} N_c\sqrt{V_c}\right)^2 \le K\sum_{c\in\mathcal{P}} N_c^2 V_c,$$

so $\Delta \ge 1$, with equality if and only if all $N_c^2 V_c$ are equal across phases. When variance concentrates on few phases such that $\max_c N_c^2 V_c \gg \frac{1}{K}\sum_c N_c^2 V_c$, then $\Delta \gg 1$: the optimal allocation yields substantially fewer convergence steps than uniform allocation at the same chunk budget B.

B.2 Bias Analysis of the Masked Estimator

The masked objective in Eq. (11) omits the N_c/b_c rescaling of the unbiased estimator (Eq. 14), trading a small bias for lower variance. We bound this bias.

Lemma 3 (Bias of the masked estimator). *Let p_c denote the keep probability for phase c. The bias of the masked estimator relative to the unbiased estimator satisfies*

$$\|\text{bias}\| = \left\|\sum_{c\in\mathcal{P}}(1-p_c)\,g_c\right\| \le \sum_{c\in\mathcal{P}}(1-p_c)\,\|g_c\|.$$

Proof. Under weighted sampling without replacement with inclusion probability p_c, the expected contribution of phase c to the masked estimator is $p_c \cdot g_c$, while the unbiased estimator yields g_c. Summing across phases and applying the triangle inequality gives the bound. □

Under the allocation $p_c \propto \sqrt{V_c}$, $(1 - p_c)$ is large only when V_c is small, and by Lemma 1, small V_c implies $\|g_c\| \to 0$. Each term in the bound is therefore suppressed by at least one small factor, so Theorem 1's allocation remains approximately optimal under the biased estimator.

C Methods We Tried That Did Not Work

The motivating observation behind this paper was that VLA rollouts on a given task are remarkably similar across trajectories. Most timesteps execute nearly identical motion patterns regardless of whether the trajectory eventually succeeds. A reaching phase looks like a reaching phase across all rollouts; the divergence concentrates on a small number of decision points. This led us to hypothesize that gradient compute was being misallocated: the same per-step budget was being spent on phases that contributed little to the policy update as on phases that determined the outcome. Before arriving at PCM, we explored two alternative directions for exploiting this observation. Both failed, and the failures shaped the eventual design.

C.1 Branching at Decision Critical Timesteps

Our initial approach attempted to increase the rollout signal at high-uncertainty points by branching: at each candidate decision point, fork the rollout into K alternative continuations to give GRPO more contrastive signal in the regions that mattered. Two problems made this infeasible. First, each branch required actually executing a downstream rollout in the simulator to be useful for GRPO, which meant returning to the branch point and continuing the simulation; this added significant rollout-side overhead and partially undid the efficiency motivation. Second, naive branching is exponential in depth, refining beyond the first branch point requires nested branching, and the rollout count grows as K^d where d is the number of branches per trajectory. We found no principled way to bound depth without re-introducing the same uniform-allocation problem we were trying to escape.

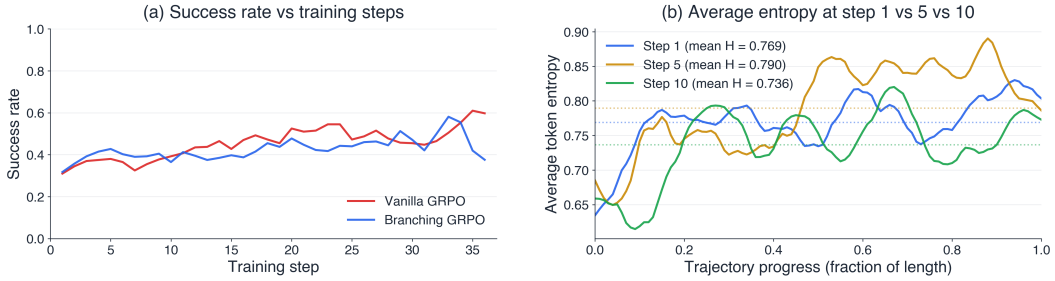


Figure 7: (a) Success vs. training steps for vanilla GRPO vs branching GRPO. (b) Action entropy over trajectory progress at different stages of training, showing weak alignment with outcome-critical phases. The training and evaluation follow the same procedures as Sec. 5.2

.

The lesson from this attempt was that any practical method had to operate within the existing rollout structure rather than expanding it. PCM instead reduces gradient computation over rollouts that have already been generated.

C.2 Entropy as a Signal to Concentrate Learning

Our second attempt used per-chunk policy entropy as a proxy for "where the model has room to learn." The hypothesis was the standard one: high-entropy chunks indicate uncertainty, uncertainty indicates a learning opportunity, and concentrating gradient on these chunks should yield faster convergence. We expected that as training progressed, entropy would decrease in initially-uncertain chunks and the gradient allocation would naturally shift toward the remaining noisy regions.

Two empirical findings led us to abandon this approach. First, entropy was poorly structured across trajectories: we did not observe consistent concentration on decision-critical phases the way we

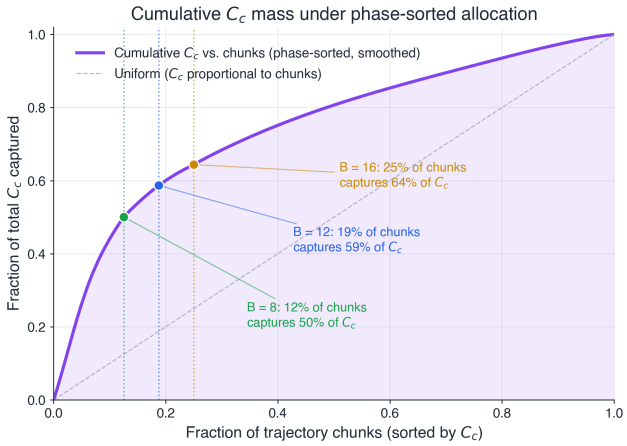


Figure 8: Cumulative C_c captured as a function of the fraction of trajectory chunks retained, with chunks sorted in descending order by phase-level C_c. The solid curve shows the empirical cumulative C_c, while the dashed diagonal shows the uniform baseline under which C_c would scale linearly with chunk count. The gap between the two reflects the uneven distribution of learning signal across the trajectory: a small fraction of chunks captures a disproportionate share of the total C_c.

eventually observed for C_c (Figure 1). Per-chunk entropy varied substantially across rollouts even within the same task and phase, and the high-entropy chunks did not align with the points where successful and failed rollouts actually diverged. We attribute this to the fact that VLA action distributions are typically already low-entropy after SFT. The supervised pretraining drives the policy toward confident, low-entropy actions, and the residual entropy reflects modeling noise rather than genuine multimodal uncertainty over correct behavior. Second, and more decisively, entropy did not meaningfully decrease over training. Across multiple runs on LIBERO, we measured per-chunk entropy at the start and end of GRPO fine-tuning and found negligible reduction. This is consistent with the known property that GRPO is a relatively weak signal for changing the underlying action distribution: the group-relative advantage primarily reweights existing modes rather than collapsing the policy onto a sharper one. Entropy under SFT-initialized policies is therefore close to a fixed property of the model rather than a quantity RL can drive down. Selecting on entropy would mean repeatedly allocating gradient to the same chunks throughout training regardless of whether they had become well-learned, and would not adapt to the shifting locus of the success–failure gap as training progressed.

D Analysis of Chunk Budget

The compute-signal trade-off. Choosing B reflects a trade-off between two opposing pressures. Lowering B reduces per-step gradient computation linearly: fewer chunks in the backward pass reduce activation memory and accelerate actor updates, which is the source of PCM's wall-clock advantage. Increasing B captures more of the available learning signal: a larger budget includes more chunks from outcome-divergent phases and approaches the gradient quality of full-trajectory GRPO.

The cumulative C_c curve in Fig. 8 is sharply concave: early chunks, corresponding to high-C_c phases, deliver disproportionate signal, while additional chunks beyond the knee contribute progressively less. The *knee* of this curve, the point at which marginal C_c capture per chunk drops sharply, occurs at approximately 20% of trajectory chunks. This concavity makes the trade-off non-trivial. Below the knee, each additional chunk yields substantial signal at fixed compute cost, so excluding it sacrifices accuracy for marginal speedup. Above the knee, each additional chunk yields diminishing signal at the same cost, so including it sacrifices speedup for marginal accuracy. The knee marks the point where these pressures balance: the smallest B for which captured C_c has largely saturated and further increases primarily incur additional compute cost.

Budget range selection. The cumulative C_c curve in Fig. 8 is sharply concave, with a knee at approximately 20% of trajectory chunks. We compute captured C_c by ranking chunks according

to their phase score and measuring the fraction of total chunk-level score mass retained by the top B chunks. Beyond the knee, each additional chunk yields progressively smaller marginal gains in captured C_c. We select $B \in \{8, 12, 16\}$ to bracket this knee, corresponding to 12%, 19%, and 25% of trajectory chunks, respectively.

- $B = 8$ (12% of chunks, 50% of C_c captured): Below the knee. Tests whether aggressive truncation that falls short of the inflection leads to signal under-coverage and degraded final accuracy.
- $B = 12$ (19% of chunks, 59% of C_c captured): At the knee. Tests the predicted operating point where C_c capture is high and marginal returns begin to flatten.
- $B = 16$ (25% of chunks, 64% of C_c captured): Beyond the knee. Tests whether additional budget yields meaningful gains. The extra 5 percentage points of C_c capture over $B = 12$ come at substantially higher per-step compute.

The empirical sweep in Sec. 5.2 (RQ3) confirms this curvature: $B = 8$ underperforms in final accuracy, $B = 16$ matches $B = 12$ but at higher cost, and $B = 12$ emerges as the operating point consistent with the knee of the cumulative C_c curve.

Choosing B for other domains. The knee of the cumulative C_c curve provides a principled approximation for B in any RL setting with a defined phase partition. The reasoning generalizes directly from the compute–signal trade-off: C_c is computed from rollouts that GRPO already produces, so the curve can be measured in any domain where outcomes are verifiable and rollouts decompose into phases. The structural assumptions of the method are therefore not specific to manipulation. The knee identifies the operating point at which the two pressures balance, independent of the absolute magnitude of C_c or the number of phases, because it is determined by the curvature of the signal.

The procedure is the same across domains. For a target domain, compute C_c per phase from a single rollout group, plot the cumulative C_c curve as in Fig. 8, and select B at the knee, the smallest fraction of chunks at which captured C_c has largely saturated. The phase partition itself is domain-specific, such as gripper-state heuristics for manipulation, semantic decomposition for LLM reasoning, or episode segmentation for dialogue. Once defined, the curve and its knee are computed identically. Domains with sharper curvature, where signal concentrates in a few phases, admit smaller B and larger speedups. Flatter curves indicate more uniform signal and require larger B to capture comparable C_c, yielding smaller speedups. Only the resulting value of B varies; the selection rule remains unchanged.

E Implementation Details

This appendix provides additional experimental details omitted from the main text, including benchmark construction, training hyperparameters, PCM-specific settings, and evaluation protocol. Unless stated otherwise, PCM and the full-trajectory GRPO baseline use the same rollout, reward, advantage, optimizer, and evaluation configuration.

E.1 Benchmarks

Models and Benchmarks. We evaluate on three LIBERO suites to cover diverse manipulation regimes and test generalization. LIBERO-Object tests object-centric pick-and-place, LIBERO-Spatial tests spatial-relation following, and LIBERO-Goal tests goal-conditioned manipulation. The manipulation tasks are specified by natural-language instructions. The following examples illustrate the types of prompts used across the benchmarks:

- Object-centric manipulation: “pick up the mug and place it in the bowl.”
- Spatial manipulation: “place the object on the left side of the tray.”
- Goal-conditioned manipulation: “move the object to the target location.”

The policy receives the language instruction and RGB observation and predicts a sequence of 7-DoF end-effector actions. These suites vary in object identity, spatial layout, goal specification, and task structure, bringing diversity in manipulation regimes and causing phase-level failure patterns to

appear with different frequencies across benchmarks. Each benchmark contains 10 tasks with 50 trials per task under different initial object configurations, yielding 500 evaluation trials per benchmark. Trajectories contain a total of 64 chunks. During RL fine-tuning, we use visual augmentations such as brightness changes and image jitter to improve robustness.

Training. Our method is implemented based on the SimpleVLA-RL `verl` pipeline [Sheng et al., 2025]. We initialize from the SimpleVLA-RL 1-trajectory SFT checkpoint, where OpenVLA-OFT is supervised-fine-tuned with one demonstration per task. We then perform RL fine-tuning with LoRA rank r=32 and α=32 on 2 NVIDIA H100 GPUs. We use a prompt batch size of 8 and sample 10 rollouts per prompt, giving 80 trajectories per training step. The trajectory mini-batch size is 4, and the PPO mini-batch size is 2 with micro-batch size 1. We optimize the actor with AdamW using a constant learning rate of 1×10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 0, and gradient clipping at 2.0. Following SimpleVLA-RL, we use one PPO epoch per update, asymmetric clipping $(\epsilon_{\text{low}}, \epsilon_{\text{high}}) = (0.2, 0.4)$, entropy coefficient 10^{-3}, and disable the KL penalty. Rollouts are sampled with temperature 1.6, top-$p = 1.0$, and top-$k = -1$ without truncation.

For PCM, we use a fixed chunk budget of B=12 chunks per trajectory. Phase keep probabilities are initialized from the success–failure phase scores C_c computed from the rollout group, and then recomputed every T_{rc}=5 training steps from the recent phase-score buffer. We clip keep probabilities below by $p_{\min}$=0.1, ensuring that low-variance phases are not completely discarded and preserving a small amount of exploration throughout training. Non-selected chunks are physically removed before the actor forward and backward pass. PCM is simple to integrate into existing GRPO pipelines: it only adds phase-score computation from rollout statistics and a chunk-shrinking step before the actor update. We use the same PCM hyperparameters across all benchmarks without task-specific tuning.

E.2 Phase Allotment Rules

We assign trajectory chunks to semantic phases using a deterministic gripper-based heuristic. For each chunk j, we compute $g_f[j] \in [0, 1]$, the fraction of timesteps in the chunk where the gripper-close command is active. We use a sustained-close threshold $\tau = 0.75$ to locate grasp intervals, and then assign phase labels around those intervals.

- **Active-grip.** Chunks with substantial gripper closure are labeled *active-grip*. We use a softer threshold $g_f[j] \geq 0.5$ for labeling, so this phase includes both sustained grasp chunks and nearby partial-close chunks. In successful rollouts, this covers grasp and transport while the object is held, or recovery after drop; in failed rollouts, it can also capture drops, re-grasp attempts, or repeated failed closure near the object.
- **Pre-grasp.** Up to three chunks immediately before a sustained-close interval are labeled *pre-grasp* when the gripper has begun closing but has not yet reached sustained closure $(0.1 \leq g_f[j] < 0.5)$. This phase often captures final alignment and orientation adjustment before contact.
- **Release-ramp.** Up to three chunks immediately after a sustained-close interval are labeled *release-ramp* $(g_f[j] < 0.5)$, capturing the transition from holding the object to releasing it. This phase is important when failures occur because the object is released at the wrong location or does not fall into the target bin.
- **Approach.** Chunks before the pre-grasp/active-grip region of a grasp cycle are labeled *approach* $(g_f[j] < 0.5)$, corresponding to navigation toward the object.
- **Tail** Chunks after the final release with an open gripper are labeled *tail* $(g_f[j] < 0.1)$. These often capture post-release behavior, including cases where the object was not successfully placed and the policy continues hovering or moving after release or attempts.

The unified phase set is resolved by priority:

$$\textit{active-grip} > \textit{pre-grasp} > \textit{release-ramp} > \textit{approach} > \textit{tail}$$

When multiple phase rules apply due to overlapping windows, we assign the highest-priority label. Phases are distinguished first by their position relative to sustained-close intervals and then by the gripper thresholds above: open-gripper chunks before a future grasp attempt are *approach*, whereas open-gripper chunks after the final release-ramp window (after a sustained grasp interval) are *tail*.

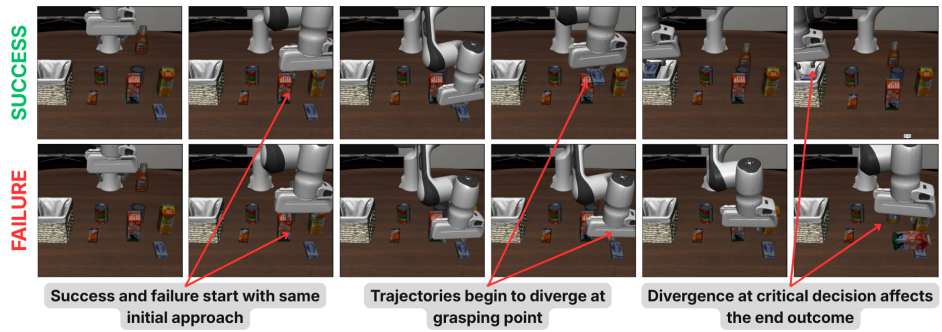


Figure 9: Successful and failed manipulation rollouts look similar for most of the trajectory, diverging mainly around outcome-critical grasp phases.

Since labels are assigned at the chunk level, averaging the gripper command over the chunk reduces sensitivity to single-timestep noise and makes the thresholding rule more stable.

The same rule is applied to successful and failed trajectories. The terminal success label is never used by the phase classifier, so C_c compares success and failure trajectories under a shared phase partition. The labeling rule is lightweight, requires no learned phase model, and adds no inference overhead. Other phase allocators, such as task-specific heuristics, temporal semantic phase segmenters, or LLM-based labelers, could be used when gripper state is insufficient.

Phase-based failure localization: Figure 9 illustrates how common failures fall naturally into this phase partition. In a missed-grasp failure, the robot may approach the object similarly to successful rollouts but close with poor alignment. The closing chunks are labeled *active-grip*, while the immediately preceding orientation-adjustment chunks are labeled *pre-grasp*. This explains why both pre-grasp and active-grip can receive high C_c: the failure may be caused not only by the closure itself, but also by the pose and alignment right before contact. After such failures, however, many remaining chunks are downstream consequence states (hovering, searching, colliding, or moving without the object) which are typically labeled *tail*. These chunks can occupy a large fraction of failed rollouts, but they are less causally informative than the contact error that produced them. PCM therefore keeps them with low nonzero probability while concentrating budget on the grasp phases that prevent the policy from entering these failed states.

Other failures are captured by the same heuristic without additional case rules. If the object is grasped but later dropped or repeatedly re-grasped during transport, the corresponding close/hold chunks remain *active-grip*, making this phase sensitive to transport instability. These chunks provide useful RL signal by penalizing slips, failed recovery and unstable holds while rewarding successful recovery. If the grasp succeeds but placement fails, the error appears around *release-ramp*: the robot releases at the wrong location or the object misses the bin, making its C_c moderately high. Subsequent *tail* chunks mostly capture open-gripper hovering after release, and are therefore less outcome-discriminative and provide little learning signal. When computing C_c, we compare only phases that contain both successful and failed samples in the rollout group; phases absent from one outcome group in a batch are skipped for that update. In the observed rollout groups, failures often diverge during active-grip, making it the most outcome-discriminative phase and causing C_c to become largest there. This concentration emerges automatically from the gripper-based phase labels and rollout outcomes, without hand-designed phase weights or task-specific tuning.