PolicyTrim: Boosting Intrinsic Policy Efficiency of Vision-Language-Action Models

Xianghui Wang[1⋆], Feng Chen[2⋆], Wenbo Zhang[2], Hua Yan[1], Zixuan Wang[1†], Changsheng Li[3], and Yinjie Lei[1‡]

[1] Sichuan University, Chengdu, China
[2] Adelaide University, Adelaide, Australia
[3] Beijing Institute of Technology, Beijing, China
wangxianghui811421084@gmail.com, yinjie@scu.edu.cn

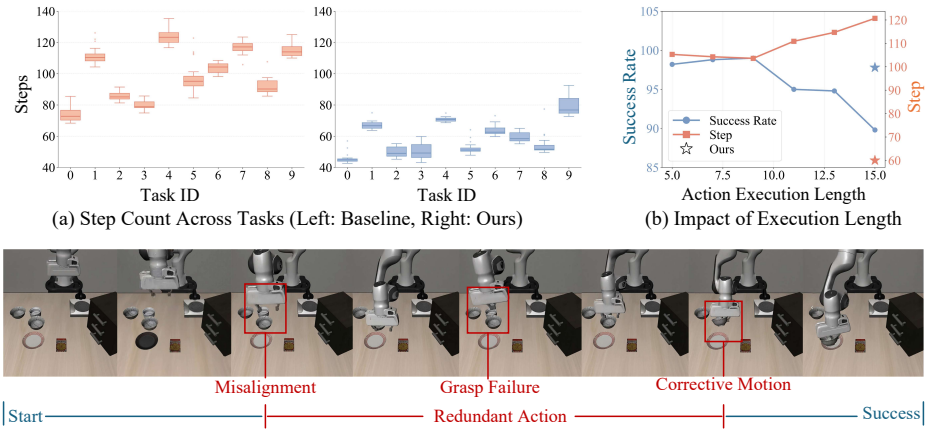
Abstract. Vision-Language-Action (VLA) models provide a unified paradigm for robotic manipulation, yet their real-world deployment remains limited by execution efficiency. While existing efforts predominantly focus on compute-centric efficiency to reduce per-step inference latency, the intrinsic **policy efficiency** of these models remains largely unexplored. Policy efficiency is fundamentally affected by two factors, namely the effective executable length of predicted action chunks and the total physical steps required to complete a task. These two factors jointly determine the number of forward inference calls during execution. We observe that current VLA policies struggle with planning unreliability and action redundancy, suffering from severe tail degradation in action chunks and tending to generate redundant physical steps. To address this, we propose **PolicyTrim**, a reinforcement learning-based post-training framework that extends reliable action chunk length and reduces redundant physical steps. For reliable chunk extension, we employ a dynamic exploration strategy that rewards the successful completion of longer executable lengths, progressively pushing the trustworthy prediction horizon to its empirical limit. For step efficiency, we design a redundancy-aware reward that favors successful task completion with fewer steps while penalizing unreproducible shortcuts, effectively eliminating redundant physical actions. Experiments on three benchmarks and three VLA models show that PolicyTrim improves action chunk utilization by 3×, reduces physical execution steps by 51.4%, and delivers up to a 5.83× end-to-end deployment speedup without compromising task success rates. **Project Page**: https://inceptionwang.github.io/PolicyTrim/

Keywords: Vision-Language-Action Models · Policy Efficiency · RL Post-training

1 Introduction

Vision-Language-Action (VLA) models integrate visual perception, language understanding, and action generation into a single end-to-end framework, establish-

⋆ Equal contribution. † Project lead. ‡ Corresponding author.



(a) Step Count Across Tasks (Left: Baseline, Right: Ours)
(b) Impact of Execution Length
(c) Visualization of Redundant Actions Induced by Extended Execution Length

Fig. 1: Intrinsic policy inefficiency in deployed VLA models manifests along two dimensions. (a) Repeated rollouts on identical tasks reveal substantial variance in step counts, indicating concise execution paths exist but emerge only by chance. (b) Forcing longer action chunk execution simultaneously degrades success rates and inflates physical steps, confirming that unreliable tail predictions are a key factor. (c) A visualization of how tail prediction errors trigger misalignment and grasp failures, compelling the robot into redundant corrective actions before eventual task completion.

ing a scalable paradigm for general-purpose robotic manipulation [2–4,10–12,19, 27]. To reduce the computational overhead of large vision-language backbones, the community has pursued compute-centric remedies such as visual token pruning [24], quantization, and KV-cache reuse [20,26,47,51,59], all aimed at reducing per-inference latency. However, the **policy efficiency** bottleneck of the models is largely unexplored, governed by the effective executable length of predicted action chunks and the total physical steps required to complete a task. These two factors jointly determine the total number of forward inference calls during execution. Empirically, current VLA models face planning unreliability and action redundancy, exhibiting severe prediction degradation at the tail of action chunks and tending to generate redundant execution steps. As illustrated in Fig. 1(b), when the model executes longer action chunks per inference, we observe a simultaneous decline in task success rates and an increase in the average physical steps required to complete the task. We believe this phenomenon stems from the degrading quality of tail predictions within action chunks [17, 43], resulting in an accumulation of physical errors that forces the robot to take redundant corrective actions. Furthermore, Fig. 1(a) shows that rolling out a trained model multiple times on the same task reveals substantial variance in successful execution lengths. This indicates that more compact and efficient execution paths are physically reachable but currently emerge only by chance, leaving ample room for optimizing the model's step efficiency. Consequently, intrinsic policy efficiency remains the primary bottleneck for deployed VLA systems.

In this paper, we propose **PolicyTrim**, a two-stage RL-based post-training framework that enhances the policy efficiency of VLA models through reliable chunk extension and redundant step reduction [36]. For reliable chunk extension, PolicyTrim diversifies the execution window lengths within a sampled group by assigning a fixed window size to each individual trajectory. This mechanism serves as a progressive reliability sweep to probe whether tail predictions at each chunk position remain trustworthy throughout actual task execution. Rollouts that successfully complete the task using longer action chunks receive higher rewards, progressively pushing the reliable planning frontier toward the empirical limit. Additionally, to achieve redundant step reduction, we design a step-saving reward that grants higher values to successful rollouts reaching the goal in fewer physical steps while a stability regularizer explicitly prevents the policy from collapsing into unreproducible shortcuts. Through these two stages, PolicyTrim improves overall policy efficiency without requiring architectural modifications or additional expert data. The main contributions of this work are summarized as follows:

- We identify policy efficiency as a critical yet overlooked deployment bottleneck for VLA models and distinguish it from pure computational efficiency by highlighting the dual challenges of unreliable tail predictions in action chunks and redundant physical execution steps.
- We propose PolicyTrim, a post-training framework that extends the reliable planning horizon and reduces step redundancy, without architectural changes or extra demonstrations.
- Experiments across multiple benchmarks and models show that PolicyTrim cuts physical execution steps in half, triples action chunk utilization, and delivers up to 5.83× end-to-end speedup without sacrificing success rate.

2 Related Work

2.1 Vision-Language-Action Models

Vision-Language-Action (VLA) models unify visual perception, language understanding, and action generation into an end-to-end framework for robotic manipulation [22, 23, 29, 35, 49]. Representative architectures include autoregressive VLAs such as OpenVLA [19] and RT-2 [4], as well as diffusion-based policies such as π_0 [2], $\pi_{0.5}$ [1], and GR00T [31]. To support high-frequency continuous control, modern VLAs adopt action chunking [9, 18, 58], predicting multiple future actions at each decision step. However, action quality often degrades toward the chunk tail, and imitation-trained policies may produce redundant trajectories, reducing real-world deployment efficiency. End-to-end execution time depends on both per-step inference latency and the number of inference calls needed for task completion. Existing work has focused mainly on the former, leaving intrinsic policy efficiency largely unexplored.

2.2 Efficient Vision-Language-Action Models

Current efficiency methods reduce per-inference cost while treating the learned policy as fixed [55]. Visual token pruning [16,24,42] and action tokenization compression [32, 46] reduce input and output overhead. Speculative decoding [45], early-exit decoding [38,39], and KV-cache reuse [20,50,51] accelerate generation. Model quantization [44, 52, 56, 57] and lightweight designs [5, 15, 37, 48] further lower hardware demands. Since these methods largely preserve the original policy distribution, they inherit intrinsic inefficiencies [53], including unreliable tail actions and redundant physical steps. When a policy requires many inference calls, per-step acceleration is bounded because total execution time depends on both latency and call frequency. In contrast, improving policy efficiency directly reduces call frequency, providing an orthogonal and multiplicative source of speedup beyond computational optimization.

2.3 Reinforcement Learning for VLA

Reinforcement learning has emerged as a post-training paradigm for improving VLA policies [8, 14, 21, 28]. Early efforts used PPO [34], but its actor-critic design introduces prohibitive memory overhead for large VLA backbones [13]. GRPO [36] removes the value model by computing group-relative advantages, making RL post-training practical for large-scale VLAs. However, existing GRPO-based VLA methods typically rely on binary success rewards [6,14,21,28], which suffer from two limitations. First, at high success rates, reward variance within the sampled group collapses, weakening advantage estimates and causing learning to stagnate. Second, binary rewards cannot distinguish shorter from longer successful executions or encourage longer reliable action chunks. PolicyTrim addresses both limitations with a step-saving reward that preserves optimization signal at high success rates and a progressive chunk exploration mechanism that pushes the trustworthy prediction frontier toward the architectural limit.

3 Method

3.1 Overview

An overview of PolicyTrim is illustrated in Fig. 2. We propose a two-stage post-training framework that extends the executable action horizon per inference and reduces the number of steps required to complete a task for VLA models. At an arbitrary decision step t, the policy π_θ processes the current visual observation o_t and language instruction l to predict a sequence of future actions $a_{t:t+H}$ in parallel, where H denotes the maximum chunk capacity. Since prediction quality degrades toward the tail of action chunks, the system is typically constrained to executing only a truncated subset of actions to avoid the accumulation of physical errors. This subset defines an execution window $h = \lfloor \gamma H \rfloor$ before the model acquires a new observation for the next planning cycle, where $\gamma \in (0, 1]$

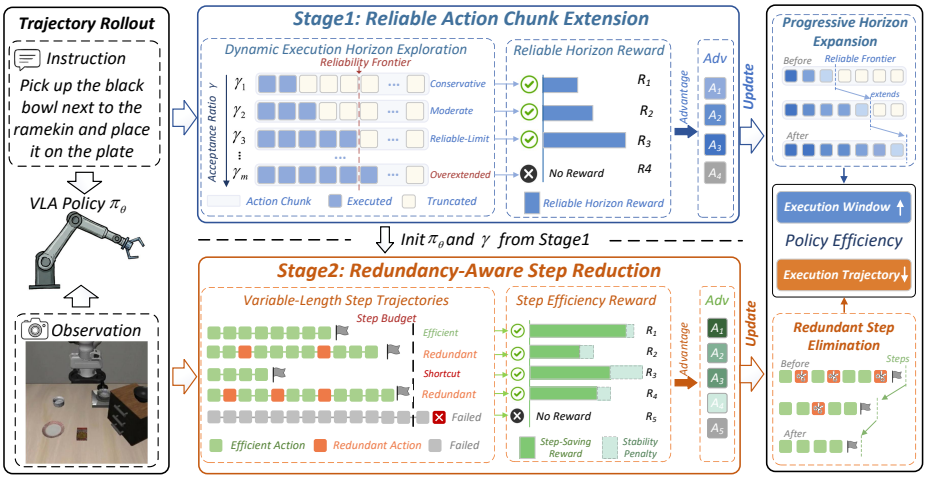


Fig. 2: Overview of PolicyTrim. PolicyTrim is a two-stage RL post-training framework that enhances intrinsic policy efficiency of VLA models. The first stage progressively extends the reliable action chunk horizon by rewarding successful execution of longer chunks. The second stage eliminates redundant physical steps via a step-saving reward coupled with group-anchored stability regularization. Together, the two stages jointly reduce the total number of forward inference calls required to complete a task.

represents the acceptance ratio. While this conservative execution strategy helps maintain physical stability, it handicaps policy efficiency by discarding potentially reliable predictions and necessitating frequent inference calls.

This paper introduces **PolicyTrim**, an RL-based post-training framework designed to enhance policy efficiency. As illustrated in Fig. 2, the framework decouples this enhancement objective into two progressive learning stages targeting reliable action chunk extension and redundancy-aware step reduction respectively. To facilitate reliable action chunk extension, the initial stage diversifies the execution lengths within a sampled group by assigning a different acceptance ratio to each individual trajectory. This mechanism implements a progressive reliability sweep that probes the empirical boundaries of trustworthy predictions. By rewarding successful trajectories proportionally to their chunk utilization, the model learns to yield more reliable action predictions further into the chunk tail and progressively extends the execution window. Building upon these extended chunks, a complementary redundancy-aware mechanism subsequently drives the second stage of physical step reduction. This phase introduces a step-saving reward coupled with a per-trajectory stability penalty. This method explicitly incentivizes the policy to prune redundant intermediate actions and converge toward the most concise physical execution trajectory while preventing catastrophic collapse onto irreproducible shortcuts. Ultimately, this sequential refinement of expanding reliable action chunks and reducing total steps naturally leads to a significant reduction in the required inference calls.

3.2 Reliable Action Chunk Extension

To enable reliable predictions over extended action horizons, the proposed method avoids rigidly forcing the model to execute the maximum length from scratch. It is instead framed as a progressive extension process where the framework dynamically probes the empirical reliability of various chunk positions by assigning execution windows of varying lengths across sampled trajectories. The policy is explicitly incentivized to overcome the inherent tail degradation by assigning higher rewards to those successful rollouts that manage to sustain longer execution length. Ultimately, this progressive refinement paradigm seamlessly pushes the trustworthy prediction frontier toward the maximum usable action chunk length H supported by the underlying model architecture.

Dynamic Execution Horizon Exploration. To probe the empirical reliability boundary without additional rollouts, each trajectory within the group is assigned a distinct execution window, collectively spanning multiple prediction horizons within a single group. Formally, we define a discrete set of acceptance ratios $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$ with $0 < \gamma_m \leq 1$, and assign each trajectory τ_i its own ratio γ_i, yielding an execution window $h_i = \lfloor \gamma_i H \rfloor$. This per-trajectory assignment turns each sampled group into a reliability sweep, probing prediction trustworthiness across short, intermediate, and long chunk positions in parallel.

Reliable Horizon Reward. The reward for chunk extension is composed of a task completion reward and a horizon reward. The task completion reward $R_{succ}(\tau_i)$ assigns 1 to a successful trajectory and 0 otherwise. The horizon reward is scaled by the assigned acceptance ratio:

$$R_{horizon}(\tau_i) = \beta \cdot \gamma_i, \tag{1}$$

where a larger γ_i indicates that the model sustained accurate predictions over a longer execution window without intermediate re-observation. However, a naive combination of the two becomes horizon-biased when no trajectory in the group succeeds, because the task completion signal collapses to a constant while the horizon reward still pushes the policy toward longer but potentially erroneous execution. To incentivize horizon extension without collapsing to unreliable long-chunk behaviors, we activate the horizon reward only when the group contains at least one successful trajectory. Concretely, the integrated reward for chunk extension is formulated as:

$$R_{ext}(\tau_i) = \mathcal{I}_{succ}^{(i)} \cdot (R_{succ}(\tau_i) + R_{horizon}(\tau_i)), \tag{2}$$

where $\mathcal{I}_{succ}^{(i)} \in \{0, 1\}$ is a binary indicator that equals 1 only if trajectory τ_i successfully completes the task.

Group-Relative Policy Update. We optimize the aforementioned objective utilizing Group Relative Policy Optimization (GRPO). For a group of G trajectories sampled from the current policy π_θ under the identical initial state o_t and language instruction l, we compute a group-normalized advantage for each trajectory based on the current objective $R_{ext}(\tau_i)$:

$$A_i = \frac{R_{ext}(\tau_i) - \mu_R}{\sigma_R + \epsilon}, \tag{3}$$

where μ_R and σ_R are the mean and standard deviation of $R_{ext}(\tau_i)$ within the group. The advantage computation then directly contrasts these varying execution lengths under the same task instance, enabling the policy to learn which chunk positions still yield trustworthy actions and where prediction quality begins to degrade. Subsequently, the policy is updated via a clipped surrogate objective with a Kullback-Leibler (KL) penalty to the reference policy π_{ref}:

$$L(\theta) = \mathbb{E}_i \left[\min\left(r_i(\theta)A_i, \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)A_i\right)\right] - \beta_{KL} D_{KL}(\pi_\theta || \pi_{ref}), \quad (4)$$

where $r_i(\theta)$ denotes the importance sampling ratio between the updated policy and the old policy distribution, and ϵ defines the clipping range that constrains $r_i(\theta)$ within $(1-\epsilon, 1+\epsilon)$ to prevent excessively large policy updates. The KL divergence $D_{KL}(\pi_\theta||\pi_{ref})$ is computed per-token across the generated action chunk to prevent substantial deviation from the manipulation priors established during pre-training.

3.3 Redundancy-Aware Step Reduction

To explicitly encourage concise execution while preserving task correctness, we introduce a step-saving reward for successful trajectories, complemented by a group-consistent step regularization to prevent exploitation of fragile shortcuts.
Step-Saving Reward. We define a step budget S_{base} based on the initial policy's performance statistics. For a successful trajectory completing the task in $S(\tau_i)$ steps, the step-saving reward is defined as:

$$R_{step}(\tau_i) = \frac{\max(0, S_{base} - S(\tau_i))}{S_{base}}. \quad (5)$$

Trajectories that complete the task in fewer steps receive proportionally higher reward, directly incentivizing the policy to converge toward more concise execution trajectories.
Group-Anchored Regularization. Naively minimizing execution steps without constraints risks policy collapse. Due to high initial variance in step counts, occasional short trajectories may receive disproportionately large rewards, causing the policy to exploit shortcuts that are not reliably reproducible. To address this, we introduce a group-anchored stability penalty that regularizes the step distribution within each sampled group:

$$P_{stab}(\tau_i) = \lambda_{stab} \cdot \tanh\left(\frac{|S(\tau_i) - \mu_{group}|}{\max(\sigma_{group},\, \sigma_{floor})}\right), \quad (6)$$

where μ_{group} and σ_{group} are the mean and standard deviation of $S(\tau)$ computed over successful trajectories in the group, serving as a group consensus anchor. The tanh function smoothly maps the normalized deviation to a bounded penalty in $[0, 1]$, ensuring that trajectories deviating substantially from the group consensus incur progressively larger penalties while avoiding unbounded penalization. This design discourages the policy from exploiting fragile step-count outliers,

guiding it to shift smoothly toward higher efficiency rather than collapsing onto irreproducible shortcuts. Meanwhile, as the policy converges to a consistent execution regime, σ_{group} may become very small, causing even minor step deviations to incur disproportionately large penalties. To address this, we introduce a floor parameter σ_{floor} to bound the denominator, preventing such over-penalization and ensuring that subsequent policy updates remain well-conditioned once a stable regime is reached.

Joint Step-Performance Reward. The final reward for step reduction integrates the task completion reward R_{succ}, the step-saving reward R_{step}, and the stability penalty P_{stab}:

$$R_{eff}(\tau_i) = \mathcal{I}_{succ}^{(i)} \cdot (R_{succ}(\tau_i) + R_{step}(\tau_i) - P_{stab}(\tau_i)), \tag{7}$$

where the binary indicator $\mathcal{I}_{succ}^{(i)}$ ensures that failed rollouts strictly receive a reward of zero, preventing step-saving incentives from reinforcing unsuccessful behaviors. We then utilize these rewards to calculate the group-relative advantage using Eq. (3) and execute the final policy update following Eq. (4).

Overall, inspired by the empirical insights from Fig. 1, we decouple the optimization into two sequential stages. Reliable chunk extension first establishes a wider trustworthy prediction horizon. However, committing to longer action chunks alone provides no mechanism to constrain the total number of physical steps, and may even inflate it through compounded tail prediction errors. Conversely, jointly optimizing both objectives simultaneously conflates two fundamentally distinct sub-goals, potentially entangling the reward signal and complicating the optimization landscape. We therefore apply redundancy-aware step reduction as a subsequent stage, building upon the extended chunk horizon to compress the physical execution path and systematically address the dual bottlenecks of VLA models.

4 Experiment

4.1 Experimental Setup

Benchmarks. We evaluate on three diverse benchmarks including LIBERO [25], ManiSkill [41], Meta-World [30] and further validate its sim-to-real transfer on a physical robot platform. Reported metrics include average success rate, average physical steps, average action chunk execution length, end-to-end execution speedup, and wall-clock execution time for real-world deployment.

- **LIBERO** is a tabletop manipulation benchmark comprising four subsets. The Spatial and Object subsets evaluate spatial reasoning and object generalization, the Goal subset introduces goal-conditioned reasoning, and the Long subset requires multi-stage manipulation over extended horizons, making it well-suited for assessing step efficiency under prolonged execution.
- **ManiSkill** is a high-fidelity simulation platform offering physics-rich continuous control tasks. Following the experimental setup in [54], we adopt its

diverse pick-and-place task combinations to evaluate policy efficiency and cross-task generalization in precise manipulation scenarios.
- **Meta-World** covers a wide range of manipulation tasks beyond pick-and-place, encompassing diverse end-effector motions and interaction modes. We adopt the MT50 suite to assess policy efficiency across heterogeneous action spaces and manipulation dynamics.
- **Real-world Deployment** uses an Agilex Piper arm equipped with two Intel RealSense D435i cameras. We evaluate three tabletop manipulation tasks, FlipMug, HangMug, and TapeBox.

VLA Models. To verify cross-architecture generalization, we apply PolicyTrim to three VLA models spanning distinct architectural paradigms. $\pi_{0.5}$ is a conditional diffusion policy that generates action chunks through iterative flow-matching denoising, conditioned on vision-language features from a pretrained VLM backbone. Its stochastic decoding naturally captures complex action distributions. OpenVLA-OFT builds upon a 7B-parameter vision-language model and replaces autoregressive token generation with parallel chunk decoding via placeholder action tokens and bidirectional attention, enabling single-forward-pass prediction of entire action sequences. GR00T is a generalist robot transformer that adopts a dual-system design, pairing a slow vision-language reasoning module with a fast diffusion-based action generation head.

Implementation Details. All experiments are built upon the RLinf framework [54]. We use a group size of $G = 8$ trajectories for each task in every iteration. For all VLA models, the maximum chunk capacity H predicted at each step is initialized to match or exceed the original settings of the respective checkpoints. For reliable chunk extension, the sampling set of the acceptance ratio is $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$ with $0 < \gamma_m \leq 1$ and $M = 3$ by default; γ_1 is set such that $\gamma_1 \cdot H$ equals the model's original default execution length, and the remaining ratios are uniformly spaced up to 1. For step efficiency, the step budget S_{base} is set to approximately 1.3 times the average successful steps of the initial baseline policy. All experiments are conducted on 8 A100 GPUs with 64 parallel simulation environments.

4.2 Main Results

Results on LIBERO. Table 1 presents a comprehensive quantitative evaluation on the four LIBERO subsets using $\pi_{0.5}$, OpenVLA-OFT, and GR00T. Across all three models and subsets, PolicyTrim consistently reduces physical execution steps while maintaining comparable task success rates, with the magnitude of reduction varying by architecture and task complexity. The most pronounced reduction is observed for $\pi_{0.5}$ on the Object subset, where the step count drops to as low as 51.4% of the baseline. These results suggest that our efficiency gains arise from a fundamentally improved execution strategy rather than any sacrifice in task competence. For action chunk extension, we observe an architectural

Table 1: Evaluation of $\pi_{0.5}$, OpenVLA-OFT, and GR00T on the four subsets of the LIBERO benchmark. We report average success rate (SR), average physical steps (S_{total}), average action chunk execution length (h_{chunk}), and end-to-end execution Speedup (Spd).

Task	Method	$\pi_{0.5}$				OpenVLA-OFT				GR00T			
		SR	S_{total}	h_{chunk}	Spd↑	SR	S_{total}	h_{chunk}	Spd↑	SR	S_{total}	h_{chunk}	Spd↑
Spatial	Baseline	97.8	108.3	5	1.0	98.6	111.2	8	1.0	91.4	67.2	5	1.0
	PolicyTrim	97.8	59.8	15	5.43	98.8	62.1	8	1.79	92.0	56.6	10	2.37
Object	Baseline	99.1	125.0	5	1.0	98.5	135.2	8	1.0	95.0	71.3	5	1.0
	PolicyTrim	98.5	64.3	15	5.83	98.5	68.8	8	1.97	95.3	65.5	10	2.18
Goal	Baseline	98.7	110.6	5	1.0	97.7	118.6	8	1.0	84.2	63.3	5	1.0
	PolicyTrim	98.8	63.5	15	5.23	98.0	66.9	8	1.77	86.3	60.8	10	2.08
Long	Baseline	93.0	249.8	5	1.0	92.9	249.3	8	1.0	86.1	177.9	5	1.0
	PolicyTrim	93.3	171.8	10	2.91	93.1	178.3	8	1.40	89.2	165.9	10	2.14

Table 2: Evaluation on ManiSkill and Meta-World. We report average success rate (SR), average physical steps (S_{total}), average action chunk execution length (h_{chunk}), and end-to-end execution Speedup (Spd).

Benchmark	Method	$\pi_{0.5}$				OpenVLA-OFT			
		SR	S_{total}	h_{chunk}	Spd↑	SR	S_{total}	h_{chunk}	Spd↑
ManiSkill	Baseline	88.1	45.2	5	1.0	60.6	53.1	8	1.0
	PolicyTrim	89.8	38.3	10	2.36	63.2	46.7	8	1.14
Meta-World	Baseline	65.1	66.3	5	1.0			–	
	PolicyTrim	65.4	52.6	10	2.52			–	

divergence among the evaluated models. $\pi_{0.5}$ and GR00T both adopt diffusion-based action decoding, which naturally accommodates extended chunk horizons; PolicyTrim successfully increases the reliable action chunk execution length for both architectures. OpenVLA-OFT, by contrast, employs a parallel decoding scheme with placeholder action tokens and bidirectional attention, where even a marginal increase in chunk length causes severe accuracy degradation, rendering chunk extension effectively untrainable for this architecture. We therefore apply only the step reduction stage to OpenVLA-OFT, and the reported chunk length improvements are specific to $\pi_{0.5}$ and GR00T.

Results on ManiSkill and Meta-World. Table 2 reports the performance on ManiSkill and Meta-World. PolicyTrim consistently improves both success rates and step efficiency across benchmarks, achieving up to 2.52× end-to-end speedup on Meta-World and 2.36× on ManiSkill with $\pi_{0.5}$. The consistent improvements across diverse physical simulators, action spaces, and model architectures demonstrate that PolicyTrim generalizes well beyond the LIBERO benchmark, confirm-

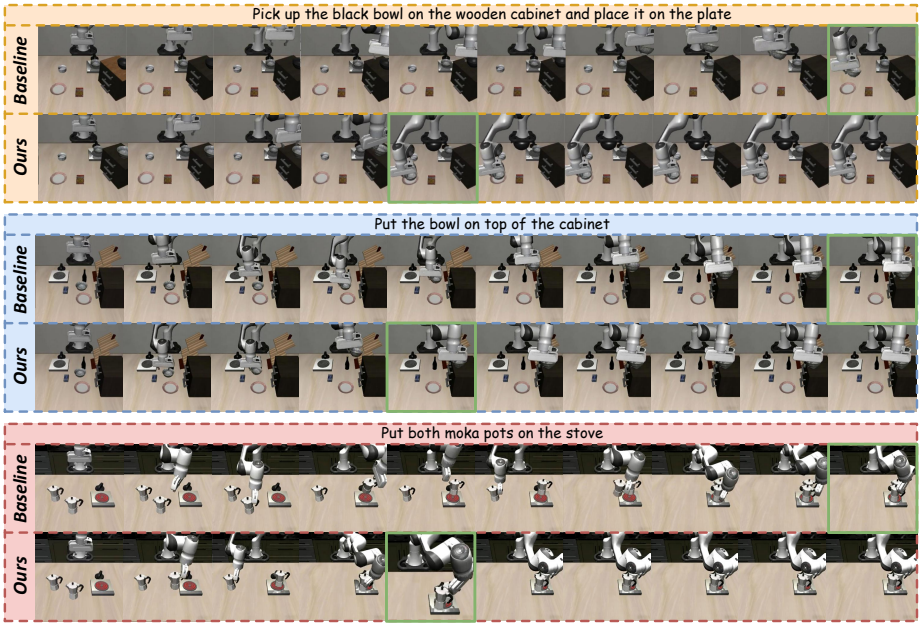


Fig. 3: Qualitative comparison on randomly sampled LIBERO tasks. Under identical configurations, the baseline incurs redundant physical actions, whereas PolicyTrim achieves task completion in roughly half the steps.

ing that policy efficiency optimization yields substantial and reliable gains across varied deployment scenarios.

Architectural Generality. Table 3 reports the cross-architecture results on both parallel-decoding and autoregressive VLA models. Beyond the standard OpenVLA-OFT setting in Table 1, we re-pretrain OpenVLA-OFT with a larger action chunk capacity of $h = 16$, since the original parallel decoder uses a fixed horizon of $h = 8$ and already executes all predicted actions, leaving no room for Stage 1 extension. With this re-pretrained OpenVLA-OFT backbone, the full two-stage PolicyTrim pipeline achieves a 2.97× speedup while preserving task success. We further evaluate the autoregressive OpenVLA model, where Stage 1 is not directly applicable because action chunks are not generated through the same fixed-horizon parallel decoding mechanism. Applying Stage 2 alone still yields a 1.41× speedup and improves the success rate from 84.7% to 87.0%. These results demonstrate that PolicyTrim provides consistent gains across different VLA decoding architectures, and that its redundancy-aware optimization is not tied to a specific action decoding form.

Real-World Deployment. Table 4 reports the real-world deployment results of $\pi_{0.5}$ on three tabletop manipulation tasks, FlipMug, HangMug, and TapeBox. PolicyTrim maintains or improves the success rate across all tasks, under both

Table 3: Cross-architecture results. We report success rate (SR), average physical steps, action horizon h, and end-to-end speedup.

Model	Method	SR	Step	h	Spd↑
OpenVLA-OFT	Baseline	98.6	111.2	8	1.00×
OpenVLA-OFT	S1+S2	98.8	65.4	14	2.97×
OpenVLA	Baseline	84.7	113.5	–	1.00×
OpenVLA	S2	87.0	80.6	–	1.41×

Table 4: Real-world deployment results. Standard uses a fixed target pose, while Dynamic perturbs the target during grasping. Values under Standard and Dynamic are success rates in %, and Time is measured in seconds.

	Standard			Dynamic		Time(Standard)		
Method	Flip	Hang	Tape	Flip	Tape	Flip	Hang	Tape
Baseline	70	60	95	70	65	14.6	15.6	17.5
PolicyTrim	75	65	95	70	70	7.6	8.7	9.4

the standard setting with a fixed target pose and the dynamic setting where the target is randomly perturbed during grasping, while substantially reducing wall-clock execution time. On average, PolicyTrim achieves a 1.86× speedup over the baseline under the standard real-world setting. These results are obtained on an Agilex Piper arm equipped with two Intel RealSense D435i cameras, where PolicyTrim is first RL post-trained in simulation and then adapted to the real world through supervised fine-tuning on a small number of real demonstrations. The consistent reduction in real execution time demonstrates that PolicyTrim's efficiency gains transfer from simulation to physical deployment.

Qualitative Analysis. Fig. 3 provides a qualitative comparison between the baseline and PolicyTrim on tasks randomly sampled from LIBERO under identical configurations. The baseline consistently exhibits redundant corrective motions before task completion, whereas PolicyTrim executes smooth and direct trajectories toward the goal. This behavioral contrast is especially pronounced in tasks involving precise placement, where the baseline displays noticeable hesitation and jittering near the target, substantially inflating the total step count. PolicyTrim avoids such corrective detours, reducing physical steps by nearly half.

4.3 Ablation Study

In Table 5, we conduct ablation studies on the LIBERO-Spatial subset using the $\pi_{0.5}$ model to isolate the contribution of each PolicyTrim component. All ablated configurations share the same initialization checkpoint and hyperparameters, with only the target component removed.

Table 5: Ablation study of different components on LIBERO-Spatial benchmarks.

Reliable Chunk Extension	Step-Saving Reward	Group-Anchored Regularization	SR	S_{total}	h_{chunk}	Spd↑
✗	✗	✗	97.8	108.3	5	1.0
✓	✗	✗	97.2	113.8	15	2.86
✗	✓	✗	93.7	81.7	5	1.32
✗	✓	✓	97.5	61.6	5	1.75
✓	✓	✓	98.3	59.8	15	5.43

Table 6: Ablation of Dynamic Execution Horizon Exploration on LIBERO-Object using $\pi_{0.5}$ with $H = 20$. Fixed-γ variants replace diverse ratio sampling with a single acceptance ratio.

Method	SR	S_{total}	h_{chunk}	Spd↑
Baseline	99.1	125.0	5	1.00
Fixed $\gamma = 0.25$	99.2	127.1	5	0.98
Fixed $\gamma = 0.50$	98.8	125.2	10	1.99
Fixed $\gamma = 0.75$	95.8	130.1	15	2.88
Fixed $\gamma = 1.0$	94.4	131.8	20	3.79
Dynamic Horizon Exploration	98.8	127.4	15	2.94

Effect of Reliable Action Chunk Extension. Adding Chunk Extension alone triples the average execution window from $h_{\text{chunk}} = 5$ to 15, reducing inference call frequency and yielding a 2.86× speedup. However, total physical steps simultaneously increase from 108.3 to 113.8, confirming that committing to longer action chunks amplifies the downstream impact of residual tail prediction errors and compels the robot to take additional corrective actions. This validates the necessity of an explicit step reduction stage following chunk extension, as longer chunks alone do not translate to more concise physical execution.

Effect of Step-Saving Reward. Applying the Step-Saving Efficiency Reward alone successfully reduces S_{total} by 24.6% from 108.3 to 81.7, demonstrating its effectiveness in driving the policy toward more concise execution. However, this comes at the cost of an unacceptable degradation in task competence, with the success rate dropping sharply from 97.8% to 93.7%. Without Group-Anchored Regularization, the policy exploits unstable short trajectories that occur by chance but are not reliably reproducible under the current policy distribution, effectively trading task success for superficial step savings. This reveals that the step reduction objective alone is insufficient, as the policy can find shorter paths but collapses onto fragile shortcuts that fail to generalize.

Table 7: Combining PolicyTrim with VLA-Cache on the four LIBERO subsets using OpenVLA-OFT. PolicyTrim and VLA-Cache target orthogonal efficiency axes and yield compounded speedups.

Method	Spatial				Object			
	SR	S_{total}	h_{chunk}	Spd↑	SR	S_{total}	h_{chunk}	Spd↑
Baseline	97.8	113.1	8	1.0	97.6	138.8	8	1.0
VLA-Cache	98.3	–	8	1.26	97.5	–	8	1.26
+PolicyTrim	98.8	63.2	8	2.26	98.5	70.5	8	2.48
Method	**Goal**				**Long**			
	SR	S_{total}	h_{chunk}	Spd↑	SR	S_{total}	h_{chunk}	Spd↑
Baseline	97.6	115.7	8	1.0	94.2	256.3	8	1.0
VLA-Cache	98.3	-	8	1.26	95.4	-	8	1.26
+PolicyTrim	98.5	65.4	8	2.23	95.4	183.1	8	1.76

Effect of Group-Anchored Regularization. When Group-Anchored Regularization is added to the Step-Saving Reward, performance improves markedly. The success rate recovers from 93.7% to 97.5% while S_{total} drops from 81.7 to 61.6. As shown in Fig. 4, without Group-Anchored Regularization, the reward collapses around step 125 as the policy exploits fragile short trajectories that cannot be reliably reproduced, causing training to destabilize. With Group-Anchored Regularization, reward instead increases steadily throughout training. The fact that a regularization term simultaneously improves both task competence and execution efficiency reveals its essential role in steering the policy away from fragile shortcuts and toward genuinely concise and reproducible execution paths.

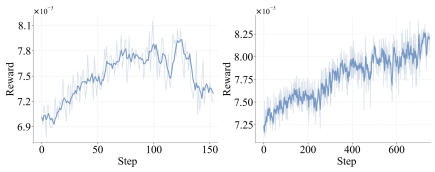


Fig. 4: Training reward curves without (Left) and with (Right) Group-Anchored Regularization on LIBERO-Spatial ($\pi_{0.5}$).

Effect of Dynamic Execution Horizon Exploration. As reported in Table 6, fixing the acceptance ratio to a single value exposes a clear trade-off: as γ increases, the execution window grows at the cost of progressive SR degradation, with Fixed $\gamma = 0.75$ and $\gamma = 1.0$ dropping to 95.8% and 94.4% respectively. Dynamic Execution Horizon Exploration resolves this tension by simultaneously probing multiple horizons within each group, achieving $h_{\text{chunk}} = 15$ comparable to Fixed $\gamma = 0.75$ while preserving an SR of 98.8% nearly identical to the baseline, underscoring its critical role in stable and effective chunk extension.

4.4 Discussion

What Causes Policy Inefficiency in VLA Policies? The root cause likely lies in the training paradigms themselves. Imitation learning [12, 27, 33] optimizes the

policy to reproduce demonstrated behaviors without any explicit signal favoring execution efficiency. Moreover, prediction errors accumulate along action chunks due to distribution shift, causing the policy to take redundant corrective actions that further inflate the total execution steps. Standard RL post-training [7, 40] with binary success rewards also provides no explicit incentive for execution efficiency. Moreover, as the policy matures and success rates rise, reward variance within each sampled group tends to collapse, gradually diminishing the discriminative power of advantage estimates.

Orthogonality with Compute-centric Methods. Policy efficiency and computational efficiency are orthogonal and jointly exploitable. While compute-centric methods reduce per-step inference latency, PolicyTrim targets the total number of forward inference calls, a dimension existing acceleration techniques leave entirely unaddressed. As demonstrated in Table 7, integrating PolicyTrim with VLA-Cache yields speedups of up to 2.48× on LIBERO-Object, well beyond the 1.26× achievable by VLA-Cache alone, confirming that the two approaches together constitute a more complete path toward practical deployment.

5 Conclusion

In this work, we identified policy efficiency as a critical yet overlooked bottleneck for deploying VLA models, and proposed **PolicyTrim**, a two-stage RL post-training framework that addresses this without architectural modifications or additional demonstrations. The first stage employs a dynamic horizon exploration mechanism to progressively push the trustworthy prediction frontier toward its empirical limit, while the second introduces a redundancy-aware reward coupled with group-anchored stability regularization to drive the policy toward genuinely concise execution without collapsing onto irreproducible shortcuts. Extensive experiments across diverse benchmarks and backbones demonstrate that PolicyTrim significantly reduces inference frequency while maintaining strong task competence. As an orthogonal complement to compute-centric acceleration, PolicyTrim can be seamlessly combined with existing inference optimization techniques for compounded efficiency gains, pointing toward a more holistic path to practical and scalable robotic deployments. While PolicyTrim provides a robust execution framework, future work may explore integrating PolicyTrim with on-device continuous learning to further adapt execution efficiency to dynamic and unseen environments.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (U23B2013, U2441242 and 62276176). This work was also partly supported by the SICHUAN Provincial Natural Science Foundation (No. 2024NSFJQ0023).

References

1. Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M.R., Finn, C., Fusai, N., Galliker, M.Y., Ghosh, D., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A.Z., Shi, L.X., Smith, L., Springenberg, J.T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H., Walling, A., Wang, H., Yu, L., Zhilinsky, U.: $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In: Lim, J., Song, S., Park, H.W. (eds.) Proceedings of The 9th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 305, pp. 17–40. PMLR (27–30 Sep 2025)
2. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L.X., Tanner, J., Vuong, Q., Walling, A., Wang, H., Zhilinsky, U.: π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164 (2024)
3. Brohan, A., Brown, N., Carbajal, J., et al.: RT-1: Robotics transformer for real-world control at scale (2022), arXiv preprint arXiv:2212.06817
4. Brohan, A., et al.: RT-2: Vision-language-action models transfer web knowledge to robotic control. In: Conference on Robot Learning (CoRL) (2023), arXiv preprint arXiv:2307.15818
5. Budzianowski, P., Maa, W., Freed, M., Mo, J., Hsiao, W., Xie, A., et al.: Edgevla: Efficient vision-language-action models (2025), arXiv preprint arXiv:2507.14049
6. Chen, F., He, Y., Lin, L., Gou, C., Liu, J., Zhuang, B., Wu, Q.: Sparsity forcing: Reinforcing token sparsity of mllms. arXiv preprint arXiv:2504.18579 (2025)
7. Chen, K., Liu, Z., Zhang, T., Guo, Z., Xu, S., Lin, H., Zang, H., Zhang, Q., Yu, Z., Fan, G., et al.: $\pi_{\texttt{RL}}$: Online RL Fine-tuning for Flow-based Vision-Language-Action Models. arXiv preprint arXiv:2510.25889 (2025)
8. Chen, Y., Tian, S., Liu, S., Zhou, Y., Li, H., Zhao, D.: ConRFT: A reinforced fine-tuning method for vla models via consistency policy. arXiv preprint arXiv:2502.05450 (2025)
9. Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. In: Robotics: Science and Systems (RSS) (2023), arXiv preprint arXiv:2303.04137
10. Collaboration, O.X.E., O'Neill, A., et al.: Open x-embodiment: Robotic learning datasets and rt-x models (2023), arXiv preprint arXiv:2310.08864
11. Driess, D., Xia, F., Sajjadi, M.S.M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al.: Palm-e: An embodied multimodal language model. In: International Conference on Machine Learning (ICML). pp. 8469–8488. PMLR (2023)
12. Ghosh, D., Walke, H.R., Pertsch, K., et al.: Octo: An open-source generalist robot policy. arXiv preprint arXiv:2405.12213 (2024)
13. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. In: International Conference on Learning Representations (ICLR) (2022)
14. Huang, D., Fang, Z., Zhang, T., Li, Y., Zhao, L., Xia, C.: Co-rft: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning (2025), arXiv preprint arXiv:2508.02219
15. Hung, C.Y., Sun, Q., Hong, P., Zadeh, A., Li, C., Tan, U., Majumder, N., Poria, S., et al.: Nora: A small open-sourced generalist vision language action model for embodied tasks. arXiv preprint arXiv:2504.19854 (2025)

16. Jiang, T., Jiang, X., Ma, Y., Wen, X., Li, B., Zhan, K., Jia, P., Liu, Y., Sun, S., Lang, X.: The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. arXiv preprint arXiv:2509.12594 (2025)
17. Jing, D., Wang, G., Liu, J., Tang, W., Sun, Z., Yao, Y., Wei, Z., Liu, Y., Lu, Z., Ding, M.: Mixture of horizons in action chunking (2025), arXiv preprint arXiv:2511.19433
18. Kim, M.J., Finn, C., Liang, P.: Fine-tuning vision-language-action models: Optimizing speed and success. arXiv preprint arXiv:2502.19645 (2025)
19. Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E.P., Sanketi, P.R., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., Finn, C.: OpenVLA: An open-source vision-language-action model. In: Agrawal, P., Kroemer, O., Burgard, W. (eds.) Proceedings of The 8th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 270, pp. 2679–2713. PMLR (06–09 Nov 2025)
20. Koo, J., Cho, T., Kang, H., Pyo, E., Oh, T.G., Kim, T., Choi, A.J.: RetoVLA: Reusing register tokens for spatial reasoning in vision-language-action models (2025), arXiv preprint arXiv:2509.21243
21. Li, H., Zuo, Y., Yu, J., Zhang, Y., et al.: Simplevla-rl: Scaling vla training via reinforcement learning (2025), arXiv preprint arXiv:2509.09674
22. Li, Q., Liang, Y., Wang, Z., et al.: Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation (2024), arXiv preprint arXiv:2411.19650
23. Li, X., Liu, M., Zhang, H., et al.: Vision-language foundation models as effective robot imitators. In: International Conference on Learning Representations (ICLR) (2024)
24. Li, Y., Meng, Y., Sun, Z., Ji, K., Tang, C., Fan, J., Ma, X., Xia, S., Wang, Z., Zhu, W.: Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration (2025), arXiv preprint arXiv:2506.12723
25. Liu, B., et al.: LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In: NeurIPS Datasets and Benchmarks Track (2023), arXiv preprint arXiv:2306.03310
26. Liu, J., Chen, H., An, P., Liu, Z., Zhang, R., Gu, C., Li, X., Guo, Z., Chen, S., Liu, M., et al.: Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. arXiv preprint arXiv:2503.10631 (2025)
27. Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., Zhu, J.: RDT-1B: a diffusion foundation model for bimanual manipulation. In: International Conference on Learning Representations (ICLR) (2025)
28. Lu, G., Guo, W., Zhang, C., Zhou, Y., Jiang, H., Gao, Z., Tang, Y., Wang, Z.: Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning (2025), arXiv preprint arXiv:2505.18719
29. Ma, Y.J., Song, Z., Zhuang, Y., Hao, J., King, I.: A survey on vision-language-action models for embodied ai (2024), arXiv preprint arXiv:2405.14093
30. McLean, R., Chatzaroulas, E., McCutcheon, L., Röder, F., Yu, T., He, Z., Zentner, K., Julian, R., Terry, J.K., Woungang, I., Farsad, N., Castro, P.S.: Meta-world+: An improved, standardized, RL benchmark. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2025)
31. NVIDIA, et al.: GR00T N1: An open foundation model for generalist humanoid robots (2025), arXiv preprint arXiv:2503.14734

32. Pertsch, K., Stachowicz, K., Ichter, B., Driess, D., Nair, S., Vuong, Q., Mees, O., Finn, C., Levine, S.: FAST: Efficient action tokenization for vision-language-action models. arXiv preprint arXiv:2501.09747 (2025)
33. Qu, D., Song, H., Chen, Q., Yao, Y., Ye, X., Ding, Y., Wang, Z., Gu, J., Zhao, B., Wang, D., Li, X.: SpatialVLA: Exploring spatial representations for visual-language-action model. arXiv preprint arXiv:2501.15830 (2025)
34. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017), arXiv preprint arXiv:1707.06347
35. Shao, R., Li, W., Zhang, L., Zhang, R., Liu, Z., Chen, R., Nie, L.: Large vlm-based vision-language-action models for robotic manipulation: A survey (2025), arXiv preprint arXiv:2508.13073
36. Shao, Z., Wang, P., Zhu, Q., et al.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models (2024), arXiv preprint arXiv:2402.03300
37. Shukor, M., Aubakirova, D., Capuano, F., Kooijmans, P., Palma, S., et al.: Smolvla: A vision-language-action model for affordable and efficient robotics (2025), arXiv preprint arXiv:2506.01844
38. Song, W., Chen, J., Ding, P., Huang, Y., Zhao, H., Wang, D., Li, H.: CEED-VLA: Consistency vision-language-action model with early-exit decoding (2025), arXiv preprint arXiv:2506.13725
39. Song, W., Chen, J., Ding, P., Zhao, H., Zhao, W., Zhong, Z., Ge, Z., Ma, J., Li, H.: PD-VLA: Accelerating vision-language-action model integrated with action chunking via parallel decoding (2025), arXiv preprint arXiv:2503.02310
40. Tan, S., Dou, K., Zhao, Y., Krähenbühl, P.: Interactive post-training for vision-language-action models (2025)
41. Tao, S., Xiang, F., Shukla, A., Qin, Y., Hinrichsen, X., Yuan, X., Bao, C., Lin, X., Liu, Y., Chan, T.k., Gao, Y., Li, X., Mu, T., Xiao, N., Gurha, A., Viswesh, N.R., Choi, Y.W., Chen, Y.R., Huang, Z., Calandra, R., Chen, R., Luo, S., Su, H.: Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. Robotics: Science and Systems (2025), arXiv preprint arXiv:2410.00425
42. Wang, H., Xu, J., Pan, J., Zhou, Y., Dai, G.: Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. arXiv preprint arXiv:2509.05614 (2025)
43. Wang, H., et al.: Vla knows its limits (2026), arXiv preprint arXiv:2602.21445
44. Wang, H., Xiong, C., Wang, R., Chen, X.: Bitvla: 1-bit vision-language-action models for robotics manipulation (2025), arXiv preprint arXiv:2506.07530
45. Wang, S., Yu, R., Yuan, Z., Yu, C., Gao, F., Wang, Y., Wong, D.F.: Spec-vla: Speculative decoding for vision-language-action models with relaxed acceptance (2025), arXiv preprint arXiv:2507.22424
46. Wang, Y., Zhu, H., Liu, M., Yang, J., Fang, H.S., He, T.: VQ-VLA: Improving vision-language-action models via scaling vector-quantized action tokenizers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 11089–11099 (October 2025)
47. Wen, J., Zhu, M., Liu, J., Liu, Z., Yang, Y., Zhang, L., Zhang, S., Zhu, Y., Xu, Y.: dvla: Diffusion vision-language-action model with multimodal chain-of-thought. arXiv preprint arXiv:2509.25681 (2025)
48. Wen, J., Zhu, Y., Li, J., Zhu, M., Tang, Z., Wu, K., Xu, Z., Liu, N., Cheng, R., Shen, C., et al.: Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. IEEE Robotics and Automation Letters (2025)
49. Wu, H., Jing, Y., Cheang, C., et al.: GR-1: Unleashing large-scale video generative pre-training for visual robot manipulation (2023), arXiv preprint arXiv:2312.13139

50. Xu, S., Wang, Y., Xia, C., Zhu, D., Huang, T., Xu, C.: Vla-cache: Efficient vision-language-action manipulation via adaptive token caching (2025), arXiv preprint arXiv:2502.02175
51. Xu, W., Zhuang, L.: Kv-efficient vla: A method of speed up vision language model with rnn-gated chunked kv cache (2025), arXiv preprint arXiv:2509.21354
52. Xu, Y., Yang, Y., Fan, Z., Liu, Y., Li, Y., Li, B., Zhang, Z.: Qvla: Not all channels are equal in vision-language-action model's quantization (2026), arXiv preprint arXiv:2602.03782
53. Yang, Y., Wang, Y., Wen, Z., Liu, Z., Zou, C., Zhang, Z., Wen, C., Zhang, L.: Efficientvla: Training-free acceleration and compression for vision-language-action models (2025), arXiv preprint arXiv:2506.10100
54. Yu, C., Wang, Y., Guo, Z., Lin, H., Xu, S., Zang, H., Zhang, Q., Wu, Y., Zhu, C., Hu, J., et al.: Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. arXiv preprint arXiv:2509.15965 (2025)
55. Yu, Z., Wang, B., Zeng, P., Zhang, H., Zhang, J., Gao, L., Song, J., Sebe, N., Shen, H.T.: A survey on efficient vision-language-action models (2025), arXiv preprint arXiv:2510.24795
56. Zhang, J., Hsieh, Y., Wang, Z., Lin, H., Wang, X., Wang, Z., Lei, Y., Zhang, M.: Quantvla: Scale-calibrated post-training quantization for vision-language-action models (2026), arXiv preprint arXiv:2602.20309
57. Zhang, Y., et al.: Sqap-vla: Saliency-aware quantization and pruning for vision-language-action models (2025), arXiv preprint arXiv:2512.08813
58. Zhao, T.Z., et al.: Learning fine-grained bimanual manipulation with low-cost hardware. In: Robotics: Science and Systems (RSS) (2023), introduces Action Chunking with Transformers (ACT)
59. Zheng, W., Li, B., Xu, B., Feng, E., Gu, J., Chen, H.: Leveraging os-level primitives for robotic action management. arXiv preprint (2025), arXiv preprint arXiv:2508.10259