PAPO-VLA: Planning-Aware Policy Optimization for Vision-Language-Action Models

Peizheng Guo[1,2]*, **Jingyao Wang**[1,2]*, **Changwen Zheng**[1,2], **Wenwen Qiang**[1,2]†
[1]National Key Laboratory of Space Integrated Information System,
Institute of Software, Chinese Academy of Sciences,
[2]University of Chinese Academy of Sciences
{guopeizheng2025, wangjingyao2023, changwen, qiangwenwen}@iscas.ac.cn

Abstract

Vision-Language-Action (VLA) models show promising ability in language-guided robotic tasks. However, making VLA policies reliable remains challenging, because a manipulation task is completed through closed-loop interaction, where each action affects subsequent execution. To analyze this problem, we revisit VLA policy during execution and argue that a VLA policy acts both as a planner, which makes task-oriented decisions that change the direction of execution, and as an executor, which realizes these decisions through dense continuous actions. This view suggests that improving VLA reliability requires particular attention to planning actions. Existing optimization methods can imitate actions or improve complete trajectories, but they usually do not explicitly identify planning actions or measure their importance for task success. To address this issue, we propose Planning-Aware Policy Optimization for VLA models (PAPO-VLA). PAPO-VLA first identifies planning actions by jointly considering action variation and trajectory outcome, then estimates their importance through causal sufficiency and causal necessity, and finally incorporates this importance into GRPO advantage estimation. In this way, more important planning actions receive stronger optimization emphasis, while the whole trajectory is still optimized by trajectory-level feedback. Experiments on multiple benchmarks demonstrate the effectiveness of PAPO-VLA. The code is available at https://github.com/ML-TASA/PAPO-VLA.

1 Introduction

Vision-Language-Action (VLA) models have shown strong potential in various language-guided robotic tasks by mapping visual observations and language instructions into executable robot actions [37, 29, 10, 2, 15]. However, making VLA policies reliable remains challenging. A manipulation task is completed through closed-loop interaction, where each action changes the environment and affects subsequent observations and decisions [15, 23]. Therefore, a reliable VLA policy should not only generate locally plausible actions, but also maintain the whole execution toward task success. This raises a central question: **how to guarantee the reliability of VLA policies in robotic tasks?**

To answer this question, we revisit VLA policy and argue that a VLA policy plays two functional roles (**Figure 1**). First, it acts as a planner when the action changes execution intention, such as switching from approaching an object to grasping it, from grasping to moving it, or from moving to releasing it. These actions determine the direction of the following execution. Second, it acts as an executor when it produces dense actions to connect, adjust, and realize these decisions under changing observations. Both roles are necessary for successful manipulation. Nevertheless, they influence

*These authors contribute equally
†Corresponding author

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

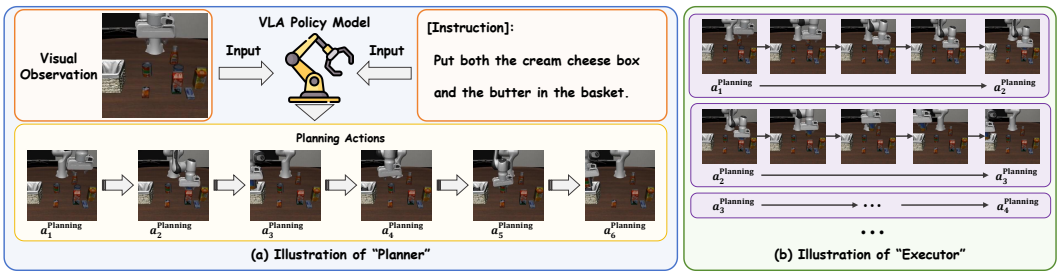


Figure 1: Illustration of Planner and Executor. (a) Planner, which consists of planning actions that are more closely related to task success; (b) Executor, which connects and realizes these planning actions through continuous motions.

reliability in different ways. Execution actions ensure motion continuity, while planning actions determine whether the following execution is built on a correct task-oriented decision. If a planning action is unreliable, such as grasping at an unstable pose or releasing before proper alignment, later execution may become difficult to recover. Thus, improving the reliability of a VLA policy requires particular attention to the reliability of its planning actions.

This perspective leads to two key problems. The first problem is how to obtain planning actions in a dense execution trajectory. In this work, we seek a way to identify planning actions from the trajectory. Since these actions often appear when the policy changes its execution intention, action variation provides a useful cue for locating planning actions [25, 12]. However, action variation alone is not sufficient. An abrupt action change may correspond to a meaningful transition, but it may also result from unstable control or a failed attempt [24, 13]. Therefore, the planning actions should also be associated with trajectory outcomes, so that the optimization focuses more on transitions that are likely to support successful execution. The second problem is how to measure the importance of these planning actions for task success. A planning action is more important if preserving it helps the subsequent execution move toward success, and if perturbing it makes the task outcome worse. Thus, we need to identify planning actions and estimate their importance for task success.

Based on this analysis, we propose Planning-Aware Policy Optimization for VLA models (PAPO-VLA). Our method first identifies planning actions by action variation and trajectory outcome. This allows policy optimization to focus on actions that indicate execution transitions and appear in trajectories with better outcomes. Then, we estimate the importance of each planning action through causal sufficiency and necessity [21]. Causal sufficiency measures whether preserving the action helps the subsequent execution reach a successful outcome. Causal necessity measures whether perturbing the action makes the outcome worse. An action receives high importance when it is both sufficient for supporting successful progress and necessary for preserving task success. Finally, we incorporate this importance into GRPO advantage estimation, so that more important planning actions receive stronger emphasis while other actions are still optimized by the trajectory outcome.

The main contributions of this paper are summarized as follows: (i) We revisit VLA policy and argue that it plays two functional roles during manipulation: a planner role that determines task-oriented execution decisions, and an executor role that realizes these decisions through dense actions. This view highlights that the reliability of planner is crucial for the reliability of VLA policy. (ii) We propose Planning-Aware Policy Optimization for VLA models (PAPO-VLA), which identifies planning actions, estimates their importance through causal sufficiency and necessity, and incorporates the resulting importance into GRPO advantage estimation; (iii) Extensive experiments conducted on various benchmarks indicate the effectiveness of our method.

2 Related Work

Vision-Language-Action Model. Vision-Language-Action (VLA) models aim to enable robots to generate executable actions in complex environments. Recent studies mainly focus on scaling robotic policies with large pretraining, and exploring post-training to adapt pretrained policies to downstream tasks. OpenVLA [10] unifies open-source vision-language modeling with robot control. $\pi_{0.7}$ [9] uses richer multimodal prompts to support manipulation across tasks, robots, and scenes. RIPT-VLA [26] shows that post-training with task-success rewards can effectively improve the adaptability of

pretrained VLA policies in new settings. TGRPO [5] introduces trajectory-wise group relative policy optimization into VLA fine-tuning, providing a direct exploration of GRPO-style post-training for VLA models. Unlike previous studies, we argue that VLA policy plays two roles during execution: planner and executor. We focus on the reliability of planner to guarantee VLA policy reliability by identify planning actions from dense trajectories, estimate their importance for task success, and incorporate this importance into GRPO-based optimization.

Causality in Robot Learning. Causal theory provides a theoretical framework for modeling interventions, counterfactual changes, and task invariance in robot learning, and has recently been used to improve policy performance. CAIAC [30] identifies state components that are not affected by actions to improve the robustness of offline robot learning. RoCoDA [1] combines causal invariance, geometric equivariance, and data augmentation to improve imitation learning under changing settings. LaVA-Man [36] uses goal-image prediction as a pretraining task to implicitly learn causal relations between visual states and actions in language-guided manipulation. These efforts show that causality can provide more structured modeling principles for robot policy learning, but mainly focus on state factors, data augmentation, or representation learning. We use causality to measure the importance of each planning action during policy optimization. By measuring causal sufficiency and necessity of such planning actions, and incorporating them into optimization, our method enables the optimization to more explicitly distinguish actions that are important for ensuring task success.

3 Problem Setting and Analysis

In this section, we first introduce the problem setting of VLA in **Section 3.1**. Then, we provide the motivation and causal analysis of VLA in **Section 3.2**. More discussion is provided in **Appendix E**.

3.1 Problem Setting

In this paper, we consider language-conditioned robotic manipulation, where a VLA receives visual observations and a task instruction, and generates an action sequence to complete the task. Specifically, let l denote the language instruction and o_t denote the visual observation at step t. At each step, the policy takes the most recent H observations and the instruction as input, and outputs an action a_t:

$$a_t \sim \pi_\theta(a_t \mid o_{t-H+1:t}, l), \tag{1}$$

where $o_{t-H+1:t} = \{o_{t-H+1}, \ldots, o_t\}$ denotes the observation window up to time step t, and θ denotes the policy parameters. This formulation covers both the case where the policy depends only on the current observation and the case where it conditions on a short history of observations.

The interaction between the policy and the environment forms a trajectory $\tau = (o_0, a_0, o_1, a_1, \ldots, o_{T_i-1}, a_{T_i-1})$, where T_i denotes the trajectory length. The corresponding action sequence is $\mathcal{A} = (a_0, a_1, \ldots, a_{T_i-1})$.

Following GRPO [6], we optimize the VLA policy by comparing multiple candidate trajectories generated under the same task condition. Specifically, given an initial input $x = (o_0, l)$, the old policy $\pi_{\theta_{\text{old}}}$ samples a group of G candidate trajectories $\{\tau^i\}_{i=1}^G$. Each trajectory τ^i is associated with a scalar trajectory-level reward r^i, which can be computed from task success, trajectory return, or their combination [6]. We optimize the policy by maximizing the following GRPO objective:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim P, \tau^i \sim \pi_{\theta_{\text{old}}}}$$
$$\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=0}^{T_i-1}\Big(\min\big(\rho_{i,t}(\theta)A_i, \text{clip}(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon)A_i\big) - \beta D_{\text{KL}}(\pi_\theta|\pi_{\text{ref}})\Big)\Big], \tag{2}$$

where P denotes the input distribution, $\text{clip}(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon)$ constrains the update magnitude with clipping threshold ϵ, and $D_{\text{KL}}(\pi_\theta|\pi_{\text{ref}})$ denotes the KL divergence between the current policy π_θ and a fixed reference policy π_{ref}. The action-level importance ratio is defined as

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(a_t^i \mid o_{t-H+1:t}^i, l)}{\pi_{\theta_{\text{old}}}(a_t^i \mid o_{t-H+1:t}^i, l)}. \tag{3}$$

The relative advantage A_i measures the relative quality of each trajectory within the group:

$$A_i = \frac{r^i - \text{mean}(r^1, \ldots, r^G)}{\text{std}(r^1, \ldots, r^G)}, \tag{4}$$

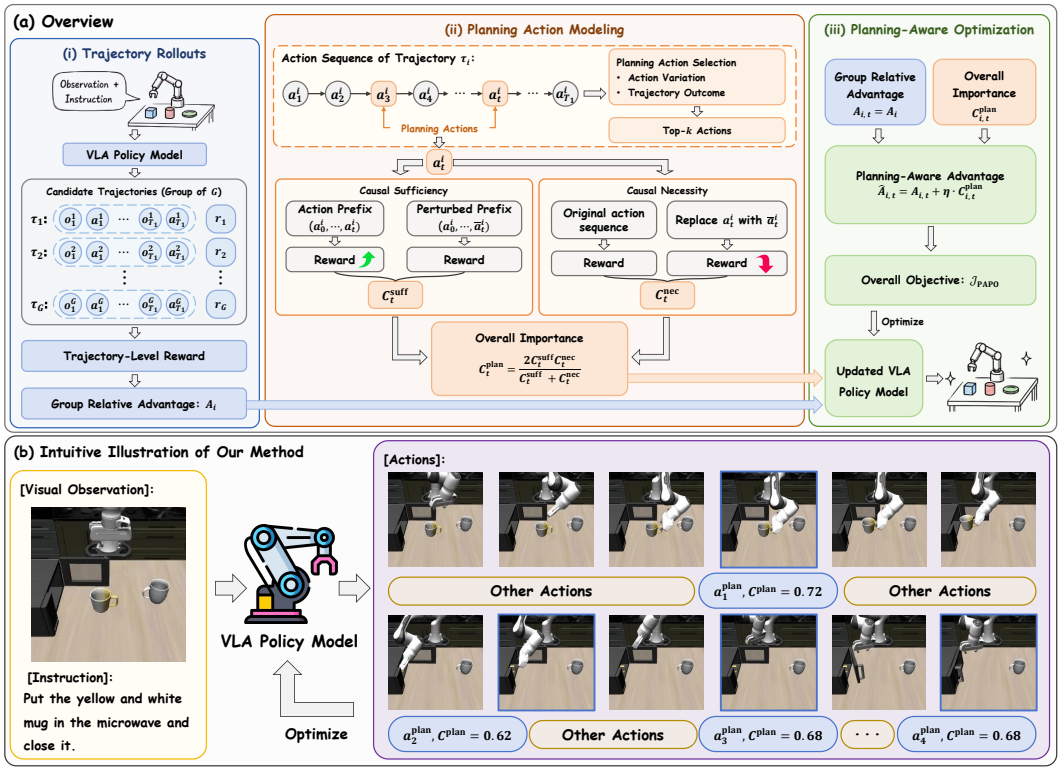


Figure 2: Overview of our proposed method. (a) Overview: Trajectory rollouts are first evaluated by trajectory-level rewards, and planning actions are selected and analyzed through causal sufficiency and necessity to obtain their importance, incorporated into the GRPO advantage estimation to guide optimization. (b) Intuitive Illustration: how different actions in a manipulation trajectory receive different importance to optimize the VLA policy.

where $\mathrm{mean}(r^1, \ldots, r^G)$ and $\mathrm{std}(r^1, \ldots, r^G)$ denote the mean and standard deviation of trajectory rewards within the same group, respectively.

3.2 Motivation Analysis

A VLA policy completes a manipulation task through a closed-loop action sequence and each action changes the environment and further affects later observations and decisions [3, 37, 10, 2, 28]. Therefore, the reliability of a VLA policy should not be understood only from whether each action is locally plausible, but from whether the whole execution can be maintained toward task success.

To analyze this problem, we revisit VLA policy during execution. Although the policy is usually formulated as a continuous action generator, it plays two functional roles in a trajectory (**Figure 1**). When the actions change execution intention, such as switching from approaching to grasping, from grasping to moving, or from moving to releasing, the policy acts as a planner. We refer to these task-oriented decision steps as planning actions. When the policy generates dense motions to connect, adjust, and realize these decisions, it acts as an executor. Both roles are necessary, but planning actions have a stronger influence on whether the following execution stays on a successful path.

This makes planning actions important for VLA reliability. If a planning action is unreliable, later execution may become ineffective even if the following actions are smooth. For example in **Figure 1**, an unstable grasp can make subsequent moving actions useless, and an early release can make the remaining trajectory difficult to recover. However, existing optimization methods usually do not explicitly distinguish such actions. Imitation learning reproduces demonstration actions, but does not indicate which actions are more important for task success. GRPO-style optimization uses trajectory-level rewards, but the same trajectory-level advantage may be assigned to all actions in a rollout, making planning actions and ordinary execution actions receive similar emphasis.

This motivates two problems. First, how can we identify planning actions from a dense execution trajectory? Since planning actions often appear when the policy changes its execution intention, action variation provides a useful cue [25, 12, 20, 27]. However, action variation alone is not sufficient, because abrupt changes may also come from unstable control or failed attempts [24, 13]. Thus, we further consider trajectory outcomes when identifying planning actions. Second, how can we measure the importance of these planning actions for task success? A planning action should be more important if preserving it helps the subsequent execution move toward success, and if perturbing it makes the task outcome worse. Therefore, we use causal sufficiency and necessity to estimate the importance of planning actions and guide policy optimization. Here, we provide the concept of probabilistic causal importance.

Definition 1 (Probabilistic Causal Importance)**.** *Let $X = (O, L)$ denote the policy input, where O is the observation and L is the language instruction, and let $\mathcal{A} = (a_1, a_2, \ldots, a_T)$ denote the action sequence generated by the VLA policy. Let $E(\mathcal{A}, X) \in \{0, 1\}$ denote the task-completion event under input X, where $E(\mathcal{A}, X) = 1$ means task success and $E(\mathcal{A}, X) = 0$ otherwise. For an action a_t, $\bar{\mathcal{A}}_t = (a_1, \ldots, \bar{a}_t, \ldots, a_T)$ denotes the perturbed action sequence by replacing the t^{th} action with a perturbed action $\bar{a}_t$. Then, the probability that a_t satisfies causal sufficiency and necessity for task success is defined as:*

$$C(a_t) = P_{\text{suff}} P\left(E(\bar{\mathcal{A}}_t, X) = 0, \bar{\mathcal{A}}_t, X\right) + P_{\text{nec}} P\left(E(\mathcal{A}, X) = 1, \mathcal{A}, X\right), \tag{5}$$

where $P_{\text{suff}} = P\left(E_{do(a_t)}(\bar{\mathcal{A}}_t, X) = 1 \mid E(\bar{\mathcal{A}}_t, X) = 0, \bar{\mathcal{A}}_t, X\right)$ denotes causal sufficiency, i.e., the probability that the task becomes successful when we intervene with $do(a_t)$ at step t, given that the perturbed sequence $\bar{\mathcal{A}}_t$ would fail. Similarly, $P_{\text{nec}} = P\left(E_{do(\bar{a}_t)}(\mathcal{A}, X) = 0 \mid E(\mathcal{A}, X) = 1, \mathcal{A}, X\right)$ denotes causal necessity, i.e., the probability that the task becomes unsuccessful when we intervene with $do(\bar{a}_t)$ at step t, given that the sequence $\mathcal{A}$ succeeds.

Definition 1 characterizes the importance of action a_t in task success. P_{suff} measures whether keeping or enforcing this action makes a previously unsuccessful execution more likely to succeed. P_{nec} measures whether perturbing or replacing this action makes a previously successful execution more likely to deteriorate. These two terms correspond to two intuitive questions: whether the action can support success, and whether the action is important for preserving success. Therefore, when an action has both high sufficiency and high necessity, it indicates that the action not only helps drive the task toward successful completion, but also has a substantial influence on the final outcome once it is changed. Such an action should be regarded as more critical to task success, and therefore should receive stronger emphasis in optimization.

4 Methods

Based on the above analysis, we propose Planning-Aware Policy Optimization for VLA models (PAPO-VLA). The core idea is to identify planning actions in trajectories, estimate their importance for task success to guide policy optimization. In **Section 4.1**, we first identify planning actions according to action variation and trajectory outcome. We then measure the importance of these planning actions from causal sufficiency and causal necessity, and aggregate them into an overall planning-action importance. In **Section 4.2**, we incorporate the planning-action importance into GRPO advantage estimation, so that actions that are more important for task success receive stronger optimization emphasis. The overview is shown in **Figure 2**, with the pseudo-code in **Appendix B**.

4.1 Planning Action Modeling

In this section, we model planning actions in a VLA trajectory. First, we identify planning actions according to action variation and task outcome (**Section 4.1.1**). Then, we evaluate their importance from two causal perspectives. Causal sufficiency measures whether preserving a planning action better supports subsequent successful execution, while causal necessity measures whether perturbing the planning action causes the task outcome to degrade (**Section 4.1.2&4.1.3**). Finally, we combine these two quantities into an overall planning-action importance (**Section 4.1.4**).

4.1.1 Planning Action Identification

Let a trajectory be written as $\tau = (o_0, a_0, o_1, a_1, \ldots, o_{T-1}, a_{T-1})$ with the corresponding action sequence $\mathcal{A} = (a_0, a_1, \ldots, a_{T-1})$. Planning actions are task-oriented decision steps in the trajectory.

They often appear when the policy changes its execution intention, such as switching from approaching to grasping or from moving to releasing. Such actions usually show larger changes compared with neighboring actions. At the same time, a large action change does not necessarily indicate a planning action, because it may also come from unstable control or a failed attempt. Therefore, we identify planning actions by jointly considering action variation and trajectory outcome.

We first compute the variation magnitude of the t^{th} action:

$$u_t = \begin{cases} \frac{1}{d} \|a_0\|_1, & t = 0, \\ \frac{1}{d} \|a_t - a_{t-1}\|_1, & t > 0, \end{cases} \tag{6}$$

where d is the action dimension, and $|\cdot|_1$ denotes the sum of absolute values over all action dimensions, so **Eq. 6** can be understood as the average magnitude of difference between the current action and the previous one across dimensions. A larger u_t indicates a more noticeable action change. To reduce the influence of action scale across trajectories, we normalize it as $\tilde{u}_t = \frac{u_t}{\frac{1}{T}\sum_{j=0}^{T-1} u_j}$.

Then, we introduce an outcome-aware gate derived from the trajectory outcome. Let $g(\tau) = \frac{r(\tau) - r_{\min}}{r_{\max} - r_{\min}} \in [0, 1]$, where $r_{\min}$ and $r_{\max}$ denote the lower and upper bounds of the trajectory reward $r(\tau)$. A larger $g(\tau)$ indicates a better task outcome. Then, the planning-action score is defined as

$$s_t = \tilde{u}_t g(\tau). \tag{7}$$

We select the top-k actions with the highest scores and denote their index set as $\mathcal{K}_\tau = \mathrm{TopK}(\{s_t\}_{t=0}^{T-1}, k)$. The corresponding planning-action mask is defined as $m_t^{\text{plan}} = \mathbb{I}[t \in \mathcal{K}_\tau]$. The top-$k$ selection involves a trade-off between candidate coverage and computational cost, with its limitations further discussed in **Appendix E.4**.

4.1.2 Causal Sufficiency Importance

According to Definition 1, causal sufficiency concerns whether keeping this action can better support the generation of a correct subsequent action sequence and eventually task success. For a planning action a_t, we define the causal sufficiency importance of action a_t as

$$C_t^{\text{suff}} = m_t^{\text{plan}} \cdot \left[\mathbb{E}_{\tau \sim \pi(\cdot|\mathcal{A}_{\leq t})} r(\tau) - \mathbb{E}_{\tau \sim \pi(\cdot|\bar{\mathcal{A}}_{\leq t})} r(\tau) \right]_+, \tag{8}$$

where $[\cdot]_+ = \max(\cdot, 0)$, $r(\tau)$ denote the trajectory-level outcome reward, $\mathcal{A}_{\leq t}$ denote the action prefix keeping the candidate action a_t, and $\bar{\mathcal{A}}_{\leq t} = (a_0, \cdots, \bar{a}_t)$ denote the perturbed prefix replacing a_t with a feasible small perturbation $\bar{a}_t$. **Eq. 8** reflects sufficiency for directly comparing the trajectory outcomes of two cases: keeping the current action versus replacing it with a perturbation. If keeping a_t leads to a higher expected reward for the subsequent trajectory, a_t provides more sufficient support for subsequent action generation and positively contributes to task success. Therefore, a larger C_t^{suff} indicates stronger causal sufficiency importance of a_t.

4.1.3 Causal Necessity Importance

Unlike causal sufficiency, causal necessity concerns whether the task outcome deteriorates significantly once the current action is replaced by another feasible action, given that the original action sequence can already support task success. To this end, for a planning action a_t, the causal necessity importance of action a_t can be defined as

$$C_t^{\text{nec}} = m_t^{\text{plan}} \cdot \left[\mathbb{E}_{\tau \sim \pi(\cdot|\mathcal{A})} r(\tau) - \mathbb{E}_{\tau \sim \pi(\cdot|\bar{\mathcal{A}}_t)} r(\tau) \right]_+. \tag{9}$$

where $[\cdot]_+ = \max(\cdot, 0)$, $r(\tau)$ denote the trajectory-level outcome reward, and $\bar{\mathcal{A}}_t = (a_0, \ldots, \bar{a}_t, \ldots, a_{T-1})$ denote the action sequence obtained by replacing the action a_t with a feasible small perturbation $\bar{a}_t$. **Eq. 9** reflects necessity for directly measuring the expected difference in trajectory outcomes before and after replacing the planning action. If replacing a_t with $\bar{a}_t$ causes a clear drop in the expected reward of the subsequent trajectory, a_t is important for maintaining the task success. Therefore, a larger C_t^{nec} indicates stronger causal necessity importance of a_t.

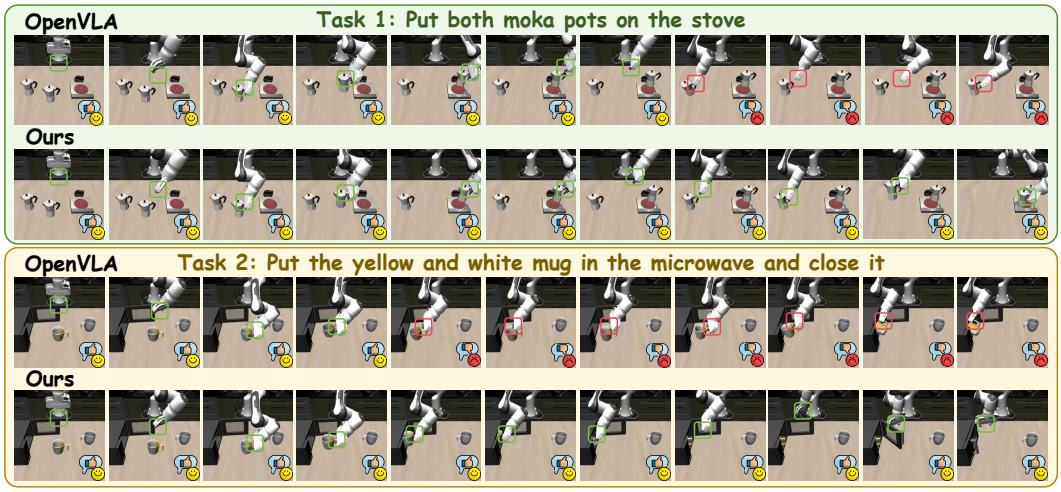


Figure 3: Visualization of comparison between OpenVLA and Ours.

4.1.4 Overall Planning-Action Importance

After obtaining the causal sufficiency importance C_t^{suff} and the causal necessity importance C_t^{nec} of action a_t, we further aggregate them into a unified quantity to characterize its overall causal importance to task success. Intuitively, an actual critical action should not only provide sufficient support for successful future behavior, but also cause noticeable degradation once it is replaced. Therefore, a planning action should be regarded as having strong importance when both sufficiency and necessity are simultaneously high. So we define the overall importance of action a_t as

$$C_t^{\text{plan}} = \frac{2C_t^{\text{suff}} C_t^{\text{nec}}}{C_t^{\text{suff}} + C_t^{\text{nec}}}. \tag{10}$$

Eq. 10 provides a characterization of the requirement of being both sufficient and necessary. When $C_t^{\text{suff}} + C_t^{\text{nec}} = 0$, we set $C_t^{\text{plan}} = 0$. If an action is high in both sufficiency and necessity, then C_t^{plan} will also be large. In contrast, if the action is high in only one aspect but weak in the other, its C_t^{plan} will be suppressed. Thus, we use C_t^{plan} as the unified planning-action importance for the policy optimization and the causal importance of non-planning positions is explicitly set to zero.

4.2 Planning-Aware Policy Optimization for VLA

In this section, we provide a feasible approach that incorporates planning-action importance into the VLA policy optimization. GRPO is a representative framework for RL, especially in settings where policy quality can be evaluated by verifiable trajectory outcomes [6]. It uses group-wise relative comparison to assess candidate trajectories, which provides a stable optimization signal without requiring additional value modeling. Therefore, we integrate the planning-action importance on the basis of GRPO for VLA optimization, constructing PAPO-VLA.

4.2.1 Planning-Aware Advantage Estimation

In **Section 3.1**, GRPO assigns each sampled trajectory τ^i a group-relative advantage A_i, which reflects the overall quality of the trajectory. A_i treats all actions within the same trajectory equally, i.e. $A_{i,t} = A_i$. We incorporate the planning-action importance obtained in **Section 4.1** into the advantage. Specifically, for action a_t^i in trajectory τ^i, let $C_{i,t}^{\text{plan}}$ denote its overall planning-action importance. We then define the planning-aware advantage as

$$\widehat{A}_{i,t} = A_{i,t} + \eta \cdot C_{i,t}^{\text{plan}}, \tag{11}$$

where η controls the strength of planning-action importance. **Eq. 11** is no longer determined only by whether a trajectory is globally good or bad, but also by planning actions.

4.2.2 Planning-Aware Policy Optimization

Based on the advantage in **Section 4.2.1**, we construct the optimization objective under the GRPO framework. Following GRPO [6], the overall objective is

$$J_{\mathrm{PAPO}}(\theta) = \mathbb{E}_{x\sim P,\tau^i\sim\pi_{\theta_{\mathrm{old}}}} \Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=0}^{T_i-1}\Big(\min\big(\rho_{i,t}(\theta)\widehat{A}_{i,t},\mathrm{clip}(\rho_{i,t}(\theta),1-\epsilon,1+\epsilon)\widehat{A}_{i,t}\big)-\beta D_{\mathrm{KL}}(\pi_\theta|\pi_{\mathrm{ref}})\Big)\Big], \tag{12}$$

where $\rho_{i,t}(\theta)$, ϵ, and β follow the same definitions as in **Section 3.1**. **Eq. 12** retains the strengths of GRPO while explicitly incorporating planning-action importance into policy learning. Actions that are both more sufficient and necessary will exert a stronger effect. The optimization no longer depends only on whether a trajectory is successful, but also focuses on planning actions that are more important for task success.

5 Experiments

In this section, we conduct extensive experiments on various benchmarks to verify the effectiveness of our method. More details, experiments, and analysis are provided in **Appendix G**.

5.1 Experimental Setup

Benchmark. We evaluate our method on two benchmarks, LIBERO [16] and RoboTwin2.0 [4], to examine its effectiveness on language-guided robotic manipulation tasks. All experiments are conducted on H100 GPU clusters. We report task success rate as the main evaluation metric.

LIBERO [16] contains four 10-task suites in the LIBERO Franka Panda simulation environment, including LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Each task provides RGB observations, robot states, task instructions, and delta end-effector actions. For evaluation, each task is tested with 50 held-out rollouts.

Table 1: Main results on the LIBERO benchmark.

Model	Spatial	Object	Goal	Long	Avg.
PackNet [19]	0.63	0.60	0.75	0.25	0.56
MTL [16]	0.83	0.54	0.80	0.48	0.66
ATM [31]	0.69	0.68	0.78	0.39	0.63
Octo [29]	0.78	0.85	0.84	0.51	0.75
OpenVLA [10]	0.85	0.88	0.79	0.53	0.76
OpenVLA-OFT [11]	0.91	0.95	0.90	0.86	0.91
TraceVLA [35]	0.85	0.85	0.75	0.54	0.75
GRAPE [34]	0.88	0.92	0.83	0.57	0.80
SFT-4LIBERO [14]	0.85	0.87	0.77	0.55	0.76
MetaVLA [14]	0.88	0.88	0.79	0.55	0.78
GRPO [6]	0.87	0.90	0.81	0.56	0.79
VLA-RL [18]	0.90	0.92	0.82	0.60	0.81
TGRPO [5]	0.90	0.92	0.81	0.59	0.81
Nora [8]	0.92	0.95	0.89	0.74	0.87
Ours	**0.93**	**0.98**	**0.98**	**0.94**	**0.96**

RoboTwin2.0 [4] contains 50 tasks with 731 object instances and introduces comprehensive domain randomization, including variations in clutter, lighting, background, tabletop height, and language instructions. In our experiments, we use the Agilex Piper robotic arm under domain-randomized task settings. Each task is evaluated on 100 held-out test scenarios. We select 12 tasks from RoboTwin2.0 and categorize them into four horizon levels according to their average step counts.

Baseline. We compare with several representative baselines: PackNet [19], MTL [16], ATM [31], Octo [29], OpenVLA [10], OpenVLA-OFT [11], RDT [17], TraceVLA [35], GRAPE [34], π_{fast} [22], Nora [8], SFT-4LIBERO [14], MetaVLA [14], GRPO [6], VLA-RL [18], and TGRPO [5].

5.2 Results Across Benchmarks

Performance on LIBERO. Table 1 presents the results on the LIBERO, showing that our method performs well across different tasks. Among them, we reach a higher success rate on LIBERO-Long, indicating that modeling planning actions can provide useful guidance for policy optimization.

Performance on RoboTwin2.0. Table 2 shows the performance on RoboTwin2.0 across short-horizon, medium-horizon, and long/extra-long-horizon task groups. Our method performs well on both short tasks and tasks that require longer and more structured execution. Additionally, PAPO-VLA shows advantages on long and extra-long tasks, indicating that planning actions modeling

Table 2: Main results of different VLA models on RoboTwin2.0, organized by task horizon.

Model	Short Horizon Tasks (100–130 Steps)				
	Lift Pot	**Beat Hammer Block**	**Pick Dual Bottles**	**Place Phone Stand**	**Avg.**
π_0 [2]	51.0	59.0	50.0	22.0	45.5
RDT [17]	45.0	22.0	18.0	13.0	24.5
OpenVLA-OFT [11]	10.1	28.1	29.7	17.1	21.3
Ours	**62.7**	**78.5**	**60.2**	**33.1**	**58.6**
Model	**Medium Horizon Tasks (150–230 Steps)**				
	Move Can Pot	**Place A2B Left**	**Place Empty Cup**	**Handover Mic**	**Avg.**
π_0 [2]	41.0	38.0	84.0	**96.0**	64.8
RDT [17]	33.0	21.0	42.0	95.0	47.8
OpenVLA-OFT [11]	28.1	37.5	77.3	45.3	47.1
Ours	**59.3**	**41.9**	**90.1**	86.7	**69.5**
Model	**Long and Extra Long Horizon Tasks (280–650 Steps)**				
	Handover Block	**Stack Bowls Two**	**Blocks Rank Rgb**	**Put Bottles Dustbin**	**Avg.**
π_0 [2]	39.0	53.0	45.0	54.0	47.8
RDT [17]	26.0	42.0	17.0	26.0	27.8
OpenVLA-OFT [11]	33.1	40.6	70.2	42.2	46.5
Ours	**50.4**	**69.9**	**77.2**	**57.8**	**63.8**

remain effective when needing to maintain correct behavior over extended action sequences. These results demonstrate the effectiveness of our method under diverse task settings and horizon lengths.

5.3 Ablation Study

The effect of different components. We examine three ablated variants: (i) Ours w/o Causal Sufficiency & Necessity (ii) Ours w/o Causal Sufficiency; (iii) Ours w/o Causal Necessity. The results in **Table 3** demonstrate the effectiveness of each component.

Table 3: The effect of planning-action importance on LIBERO.

Model	**Spatial**	**Object**	**Goal**	**Long**	**Avg.**
OpenVLA	0.85	0.88	0.79	0.53	0.76
Ours w/o Suff.&Nec.	0.87	0.90	0.81	0.56	0.79
Ours w/o Suff.	0.89	0.90	0.87	0.80	0.87
Ours w/o Nec.	0.88	0.92	0.89	0.85	0.89
Ours	**0.93**	**0.98**	**0.98**	**0.94**	**0.96**

Parameter sensitivity. We conduct experiments on the hyperparameter η and search η over $[0, 0.5]$. **Figure 6c** shows that the optimal result is at $\eta = 0.15$, which are our final configuration.

5.4 Case Study and Visualization

To further illustrate the effectiveness, we visualize two representative tasks in **Figure 3**: "putting both moka pots on the stove" and "putting the yellow and white mug in the microwave and closing it". Our method can produce more complete trajectories and maintain task progress over multiple stages.

6 Conclusion

This paper studies how to improve the reliability of VLA policies in robotic tasks. We revisit VLA execution and argue that a VLA policy plays two functional roles: a planner role that makes task-oriented decisions, and an executor role that realizes these decisions through dense continuous actions. Based on this view, we focus on planning actions, since they determine whether the following execution can remain on a successful path. We propose Planning-Aware Policy Optimization for VLA models (PAPO-VLA), which identifies planning actions from action variation and trajectory outcome, estimates their importance through causal sufficiency and causal necessity, and incorporates the resulting importance into GRPO advantage estimation. In this way, PAPO-VLA preserves trajectory-level optimization while assigning stronger emphasis to planning actions that are more important for task success. Extensive experiments demonstrate the effectiveness of PAPO-VLA.

Acknowledgments and Disclosure of Funding

The authors sincerely thank the anonymous reviewers for their valuable comments and constructive feedback. This work was supported by the National Natural Science Foundation of China under Grant No. 62506355.

References

[1] Ezra Ameperosa, Jeremy A Collins, Mrinal Jain, and Animesh Garg. Rocoda: Counterfactual data augmentation for data-efficient robot learning from demonstrations. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13250–13256. IEEE, 2025.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[4] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

[5] 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. *arXiv preprint arXiv:2506.08440*, 2025.

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

[7] Peizheng Guo, Jingyao Wang, Wenwen Qiang, Jiahuan Zhou, Changwen Zheng, and Gang Hua. Copo: Causal-oriented policy optimization for hallucinations of mllms. *arXiv preprint arXiv:2508.04182*, 2025.

[8] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

[9] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

[10] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[11] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[12] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31 (3):360–375, 2012.

[13] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, 22 (30):1–82, 2021.

[14] Chen Li, Zhantao Yang, Han Zhang, Fangyi Chen, Chenchen Zhu, Anudeepsekhar Bolimera, and Marios Savvides. Metavla: Unified meta co-training for efficient embodied adaption. *arXiv preprint arXiv:2510.05580*, 2025.

[15] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025.

[16] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[17] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[18] 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. *arXiv preprint arXiv:2505.18719*, 2025.

[19] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[20] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[21] Judea Pearl. *Causality*. Cambridge university press, 2009.

[22] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[23] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[24] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[25] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

[26] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

[27] Daniel Tanneberg, Kai Ploeger, Elmar Rueckert, and Jan Peters. Skid raw: Skill discovery from raw trajectories. *IEEE robotics and automation letters*, 6(3):4696–4703, 2021.

[28] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[29] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[30] Núria Armengol Urpí, Marco Bagatella, Marin Vlastelica, and Georg Martius. Causal action influence aware counterfactual data augmentation. *arXiv preprint arXiv:2405.18917*, 2024.

[31] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.

[32] Mengyue Yang, Zhen Fang, Yonggang Zhang, Yali Du, Furui Liu, Jean-Francois Ton, Jianhong Wang, and Jun Wang. Invariant learning via probability of sufficient and necessary causes. *Advances in Neural Information Processing Systems*, 36:79832–79857, 2023.

[33] Xiangning Yu, Zhuohan Wang, Linyi Yang, Haoxuan Li, Anjie Liu, Xiao Xue, Jun Wang, and Mengyue Yang. Causal sufficiency and necessity improves chain-of-thought reasoning. *arXiv preprint arXiv:2506.09853*, 2025.

[34] Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.

[35] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[36] Chaoran Zhu, Hengyi Wang, Yik Lung Pang, and Changjae Oh. Lava-man: Learning visual action representations for robot manipulation. *arXiv preprint arXiv:2508.19391*, 2025.

[37] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

Appendix

The appendix is organized into several sections:

- **Appendix A** provides details for all notations used in this paper.
- **Appendix B** provides the pseudo-code of our method.
- **Appendix C** provides details for the benchmarks used in the experiments.
- **Appendix D** provides details for the baselines mentioned in the main text.
- **Appendix E** provides more discussion about the proposed methodology.
- **Appendix F** contains details for the implementation of the experiment.
- **Appendix G** provides the full results and analyses of the experiment.

A Notations

In this section, we briefly describe the symbols that we mainly use in this article. We give the definitions of notation according to their role.

□ **Notations of Task and Input**

- l: the language instruction for a manipulation task.
- o_t: the visual observation at time step t.
- O: the observation sequence or observation information of the policy input.
- L: the language instruction variable in the causal formulation.
- $X = (O, L)$: the policy input, consisting of visual observation information and language instruction.
- $x = (o_0, l)$: the initial task condition used for trajectory sampling.
- H: the length of the observation window used by the policy.
- $o_{t-H+1:t} = \{o_{t-H+1}, \ldots, o_t\}$: the observation window up to time step t.

□ **Notations of Policy and Trajectory**

- π_θ: the VLA policy parameterized by θ.
- θ: the trainable parameters of the VLA policy.
- $\pi_{\theta_{\text{old}}}$: the old policy used to sample trajectories.
- π_{ref}: the fixed reference policy used for KL regularization.
- a_t: the action generated by the policy at time step t.
- $a_t \sim \pi_\theta(a_t \mid o_{t-H+1:t}, l)$: the action generation process conditioned on recent observations and the instruction.
- $\tau = (o_0, a_0, o_1, a_1, \ldots, o_{T-1}, a_{T-1})$: a trajectory generated by policy-environment interaction.
- T: the trajectory length.
- T_i: the length of the i-th sampled trajectory.
- $\mathcal{A} = (a_0, a_1, \ldots, a_{T-1})$: the action sequence corresponding to trajectory τ.
- $\mathcal{A}_{\leq t}$: the action prefix that preserves the current action a_t.
- $\bar{\mathcal{A}}_{\leq t}$: the perturbed action prefix obtained by replacing a_t with $\bar{a}_t$.
- $\bar{a}_t$: a feasible perturbed action at time step t.
- $\bar{\mathcal{A}}_t = (a_0, \ldots, \bar{a}_t, \ldots, a_{T-1})$: the perturbed action sequence obtained by replacing a_t with $\bar{a}_t$.
- $\{\tau^i\}_{i=1}^G$: a group of G trajectories sampled under the same task condition.
- G: the number of sampled trajectories in a group.

□ **Notations of Trajectory-Level Optimization**

- $r(\tau)$: the trajectory-level reward of trajectory τ.
- r^i: the trajectory-level reward of the i-th sampled trajectory τ^i.

- $\mathcal{P}$: the input distribution over task conditions.
- A_i: the group-relative advantage of the i-th trajectory.
- $\mathrm{mean}(r^1, \ldots, r^G)$: the mean reward of all sampled trajectories within the same group.
- $\mathrm{std}(r^1, \ldots, r^G)$: the standard deviation of trajectory rewards within the same group.
- $\rho_{i,t}(\theta)$: the action-level importance ratio for action a_t^i in trajectory τ^i.
- ϵ: the clipping threshold used to constrain the policy update magnitude.
- β: the coefficient of the KL regularization term.
- $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$: the KL divergence between the current policy and the reference policy.
- $J_{\mathrm{GRPO}}(\theta)$: the original GRPO objective based on trajectory-level advantages.

□ **Notations of Planning Action Identification**

- $\mathcal{K}_\tau$: the index set of planning actions identified from trajectory τ.
- $t \in \mathcal{K}_\tau$: the time step whose action a_t is identified as a planning action.
- d: the dimension of the action vector.
- u_t: the action variation magnitude of the t-th action.
- $\| \cdot \|_1$: the ℓ_1 norm over action dimensions.
- $\tilde{u}_t$: the normalized action variation score at time step t.
- $r_{\min}, r_{\max}$: the lower and upper bounds of the trajectory-level reward.
- $g(\tau)$: the outcome-aware gate derived from the trajectory-level reward.
- s_t: the planning-action score for the t-th action.
- k: the number of selected planning actions.
- m_t^{plan}: the planning-action mask at time step t.
- $\mathrm{TopK}(\{s_t\}_{t=0}^{T-1}, k)$: the operation that selects the indices of the top-k actions according to their scores.

□ **Notations of Planning Action Importance**

- $E(\mathcal{A}, X) \in \{0, 1\}$: the task-success event under input X and action sequence $\mathcal{A}$.
- $E(\mathcal{A}, X) = 1$: the task succeeds.
- $E(\mathcal{A}, X) = 0$: the task fails.
- $\mathrm{do}(a_t)$: the intervention that enforces or preserves action a_t at time step t.
- $\mathrm{do}(\bar{a}_t)$: the intervention that replaces action a_t with the perturbed action $\bar{a}_t$.
- P_{suff}: the probability of causal sufficiency, measuring whether preserving a_t can make the subsequent execution more likely to succeed.
- P_{nec}: the probability of causal necessity, measuring whether perturbing a_t can make an originally successful execution more likely to fail.
- $C(a_t)$: the probabilistic causal importance of action a_t defined from causal sufficiency and necessity.
- C_t^{suff}: the causal sufficiency importance of action a_t.
- C_t^{nec}: the causal necessity importance of action a_t.
- C_t^{plan}: the overall planning-action importance of action a_t.
- $C_{i,t}^{\mathrm{plan}}$: the overall planning-action importance of action a_t^i in trajectory τ^i.
- $[\cdot]_+$: the positive part function, defined as $[z]_+ = \max(z, 0)$.

□ **Notations of Planning-Aware Optimization**

- η: the coefficient that controls the strength of planning-action importance in advantage estimation.
- $\widehat{A}_{i,t}$: the planning-aware advantage for action a_t^i.
- $\widehat{A}_{i,t} = A_i + \eta C_{i,t}^{\mathrm{plan}}$: the modified advantage that combines trajectory-level relative advantage and planning-action importance.
- $J_{\mathrm{PAPO}}(\theta)$: the planning-aware policy optimization objective used in PAPO-VLA.

Algorithm 1 PAPO-VLA: Planning-Aware Policy Optimization for VLA

Require: Initial policy π_θ, reference policy π_{ref}, task distribution $\mathcal{P}$, group size G, number of planning actions k, planning coefficient η, clipping threshold ϵ, KL coefficient β, number of training iterations N.

Ensure: Optimized VLA policy π_θ.

for $n = 1$ to N **do**
  Sample a task condition $x = (o_0, l) \sim \mathcal{P}$.
  Set the old policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$.
  for $i = 1$ to G **do**
    Roll out a trajectory $\tau^i = (o_0^i, a_0^i, \ldots, o_{T_i-1}^i, a_{T_i-1}^i)$ using $\pi_{\theta_{\text{old}}}$ under x.
    Obtain the trajectory-level reward $r^i = r(\tau^i)$.
  end for
  Compute the group reward mean $\mu_r = \text{mean}(r^1, \ldots, r^G)$.
  Compute the group reward standard deviation $\sigma_r = \text{std}(r^1, \ldots, r^G)$.
  for $i = 1$ to G **do**
    Compute the group-relative advantage:

$$A_i = \frac{r^i - \mu_r}{\sigma_r}.$$

    Let $\mathcal{A}^i = (a_0^i, a_1^i, \ldots, a_{T_i-1}^i)$ denote the action sequence of τ^i.
    Identify planning actions according to action variation and trajectory outcome.
    Compute planning-action importance according to causal sufficiency, causal necessity, and their aggregation.
    for $t = 0$ to $T_i - 1$ **do**
      Obtain the overall planning-action importance $C_{i,t}^{\text{plan}}$.
      Compute the planning-aware advantage:

$$\widehat{A}_{i,t} = A_i + \eta C_{i,t}^{\text{plan}}.$$

      Compute the importance ratio:

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(a_t^i \mid o_{t-H+1:t}^i, l)}{\pi_{\theta_{\text{old}}}(a_t^i \mid o_{t-H+1:t}^i, l)}.$$

    end for
  end for
  Construct the PAPO objective $J_{\text{PAPO}}(\theta)$ with the planning-aware advantage.
  Update policy parameters θ by maximizing $J_{\text{PAPO}}(\theta)$.
end for
return π_θ.

B Pseudo-code

In this section, we provide the pseudo-code of PAPO-VLA. The overall procedure is shown in Algorithm 1.

C Benchmarks

In this section, we provide a brief overview of the benchmarks used in our experiments.

- LIBERO [16] is a language-conditioned robotic manipulation benchmark designed to evaluate instruction-following manipulation policies. In our experiments, we use four standard 10-task suites in the LIBERO Franka Panda simulation environment, including LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. These suites cover different types of manipulation abilities, such as spatial reasoning, object-centric manipulation, goal-conditioned execution, and longer-horizon task completion. Each task provides RGB observations, robot states, language instructions, and delta end-effector actions. Following

the standard evaluation protocol, each task is evaluated with 50 held-out rollouts, and the task success rate is used as the evaluation metric.

- RoboTwin2.0 [4] is a large-scale simulation benchmark for robotic manipulation with diverse tasks, object instances, and domain variations. It contains 50 tasks and 731 object instances, and introduces comprehensive domain randomization over clutter, lighting, background, tabletop height, and language instructions. In our experiments, we use the Agilex Piper robotic arm under the domain-randomized task settings. Each task is evaluated on 100 held-out test scenarios, and success rate is reported as the main metric. To evaluate the method under different execution lengths, we select 12 tasks from RoboTwin2.0 and group them according to their average horizon lengths, covering short-horizon, medium-horizon, and long/extra-long-horizon manipulation tasks.

D Baselines

In this section, we provide a brief overview of the baselines used for comparison in our experiments.

- PackNet [19] is a continual learning method that assigns different parameter subsets to different tasks by iterative pruning. In the LIBERO setting, it serves as a classical task-incremental learning baseline, where the model attempts to preserve previously learned task knowledge while adapting to new manipulation tasks.
- MTL [16] denotes the multi-task learning baseline used in the LIBERO benchmark. It trains a single policy jointly over multiple manipulation tasks and evaluates whether shared training across tasks can improve generalization across different task suites. This baseline provides a standard reference for comparing methods under a shared policy learning setting.
- ATM [31] models manipulation behaviors through trajectory-based policy learning. It is used as a representative imitation-learning-based manipulation baseline, where the policy learns from demonstration trajectories and predicts actions for completing language-conditioned manipulation tasks.
- Octo [29] is an open-source generalist robot policy trained on large-scale robot data. It aims to support diverse manipulation tasks with a unified policy architecture and serves as a representative large-scale generalist robotic policy baseline.
- OpenVLA [10] is an open-source Vision-Language-Action model that connects vision-language modeling with robot action generation. It takes visual observations and language instructions as input and directly predicts robot actions, making it a representative baseline for general VLA policy learning.
- OpenVLA-OFT [11] is a fine-tuned variant of OpenVLA that adapts the pretrained VLA policy to downstream manipulation tasks. It is used as an important task-adaptive fine-tuning baseline for evaluating whether the proposed optimization method can further improve VLA policy performance.
- RDT [17] is a diffusion-based robot foundation model for manipulation. It models robot actions through a diffusion policy formulation and is included as a representative baseline for evaluating performance on RoboTwin2.0, especially under diverse task settings and domain-randomized environments.
- TraceVLA [35] improves VLA policies by introducing visual trace prompting to enhance spatial-temporal awareness. It is used as a representative method that strengthens the model's ability to understand action trajectories and spatial relationships during manipulation.
- GRAPE [34] improves robot policy generalization through preference alignment. It provides a representative preference-based policy optimization baseline and is included to compare with methods that improve VLA performance through additional alignment or post-training signals.
- π_{fast} [22] improves VLA action generation by introducing efficient action tokenization. It represents a line of work that focuses on action representation design, and is used as a baseline to compare with methods that improve VLA policies through better action modeling.
- Nora [8] is a small open-source generalist Vision-Language-Action model for embodied tasks. It is included as a recent VLA baseline that aims to achieve competitive manipulation performance with a relatively compact model design.

- SFT-4LIBERO [14] denotes a supervised fine-tuning baseline on the LIBERO task suites. It adapts a pretrained VLA model to the target manipulation benchmark using demonstration data and provides a direct reference for evaluating the effect of supervised task adaptation.
- MetaVLA [14] is a meta co-training framework for efficient embodied adaptation. It aims to improve the adaptation ability of VLA models across manipulation tasks and is used as a recent fine-tuning-oriented baseline on LIBERO.
- TGRPO [5] applies trajectory-wise group relative policy optimization to VLA fine-tuning. It uses trajectory-level rewards to optimize VLA policies and is closely related to our setting.

E Discussions

E.1 Comparison and Uniqueness Discussion

In this section, we discuss the differences between PAPO-VLA and prior studies from the perspectives of research problem, task setting, and method design.

Several recent studies explore causal modeling in robot learning. RoCoDA [1] constructs counterfactual training samples to improve robot learning from demonstrations. CAIAC [30] uses causal assumptions to generate counterfactual data by modifying state components, aiming to improve policy robustness under distribution shifts. LaVA-Man [36] learns visual action representations for robot manipulation by using visual prediction as a pretraining signal. These methods show that causal modeling can provide useful principles for robot learning, but they mainly focus on data construction, state-level changes, visual representations, or pretraining objectives. In contrast, PAPO-VLA studies a different problem: during VLA policy optimization, how can we identify planning actions in dense execution trajectories, measure their importance for task success, and use this importance to guide policy updates? Therefore, our method focuses on planning-action modeling during optimization, rather than constructing counterfactual data or learning visual representations.

Causal sufficiency and necessity have also been used in feature learning and language-based reasoning tasks. PNS-based invariant learning [32] uses the probability of sufficient and necessary causes to characterize causal features for out-of-distribution generalization. Causal sufficiency and necessity for CoT reasoning [33] studies whether intermediate reasoning steps are sufficient or necessary for producing the final answer. COPO [7] estimates token-level causal contribution in MLLMs and incorporates it into GRPO-style optimization for hallucination mitigation. These works mainly consider features, reasoning steps, or generated tokens, where the input is usually fixed during prediction or generation. In contrast, PAPO-VLA is built for closed-loop robotic manipulation. A robot action is an executable operation in the environment: it changes the physical state and further affects later observations and actions. Therefore, the importance of a planning action should be evaluated through subsequent rollout and task success, rather than only through the correctness of a static prediction or generated answer.

Recent VLA post-training methods use reinforcement learning to adapt pretrained VLA policies to downstream manipulation tasks. RIPT-VLA [26] performs interactive post-training with task-success feedback. TGRPO [5] introduces trajectory-wise group relative policy optimization for VLA fine-tuning. SimpleVLA-RL [15] studies scalable VLA reinforcement learning with trajectory-level rewards. These methods show how trajectory-level feedback can be used to improve VLA policies. However, trajectory-level feedback usually provides a coarse signal at the action level. When the same trajectory-level advantage is assigned to all actions in a rollout, the optimization may not explicitly distinguish planning actions from ordinary execution actions. PAPO-VLA keeps the trajectory-level comparison in GRPO, but further introduces planning-action importance into advantage estimation. This design does not replace trajectory-level feedback. Instead, it adds an action-level term to emphasize planning actions that are more important for task success, while other execution actions still benefit from the trajectory-level learning signal.

Some VLA systems may also produce high-level plans, subtask descriptions, or intermediate reasoning steps. PAPO-VLA is different from these planning-oriented interfaces. We do not assume that the policy exposes explicit subtask outputs, nor do we introduce an additional planner. Instead, we use the planner/executor view as an operational perspective for analyzing dense action trajectories. The planning actions considered in PAPO-VLA are still low-level actions generated by the VLA policy. They are identified from action variation and trajectory outcome, and their importance is further

measured by causal sufficiency and necessity. This makes the method applicable to VLA policies that are optimized through continuous action sequences and trajectory-level rewards.

Overall, PAPO-VLA is different from existing studies in both setting and optimization target. Causal robot learning methods mainly focus on data, state factors, visual representations, or pretraining objectives. Causal sufficiency and necessity methods in feature learning and LLM/MLLM reasoning mainly focus on features, reasoning steps, or tokens under static prediction or generation settings. Existing VLA reinforcement learning methods mainly rely on trajectory-level feedback. PAPO-VLA connects causal sufficiency and necessity with planning-aware VLA policy optimization by identifying planning actions under closed-loop interaction, estimating their importance for task success, and incorporating the resulting importance into GRPO advantage estimation.

E.2 More Illustration for Motivation

In this section, we provide a more detailed explanation of the motivation behind PAPO-VLA. As shown in **Figure 1**, a VLA policy completes a manipulation task through a continuous action sequence. Although all actions are generated by the same policy, they may play different functional roles during execution. Some actions mainly connect neighboring states, adjust the pose, or maintain motion continuity. Other actions change the direction of execution, such as switching from approaching an object to grasping it, from grasping it to moving it, or from moving it to releasing it. We refer to these task-oriented decision steps as planning actions. They are still low-level actions in the executed trajectory, rather than manually annotated subtask labels or outputs of an additional planner.

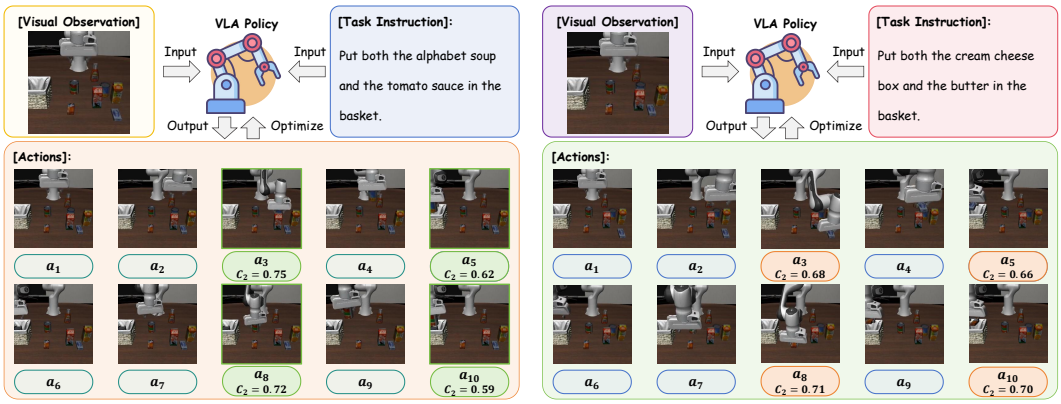
Figure 1 illustrates this idea with the task "Put both the cream cheese box and the butter in the basket". To complete this task, the robot needs to interact with two objects and place both of them into the basket. During this process, several actions have stronger influence on the direction of later execution. For example, the policy needs to reach a graspable pose near the cream cheese box, form a stable grasp, move the object toward the basket, release it at a proper location, and then perform similar decisions for the butter. These actions are planning actions because they determine what the following dense motions should realize.

At the same time, the remaining actions are also necessary. Between two planning actions, the policy still needs to generate continuous motions to move the end-effector, adjust the pose, maintain contact, and connect one execution stage to the next. These actions correspond to the executor role of the policy. For example, after grasping the cream cheese box, the robot may execute many intermediate actions to move it toward the basket. These motions are required for executable manipulation, but they usually serve to realize an existing decision. In contrast, planning actions such as grasping and releasing more directly affect whether the task can remain on a successful path.

This distinction is important for understanding VLA reliability. If the robot fails to grasp the cream cheese box stably, later motions toward the basket may no longer support task success. If the robot releases the object before it is properly aligned with the basket, the remaining trajectory may become difficult to recover. These examples suggest that errors around planning actions can propagate to later execution. Therefore, VLA optimization should not only improve the trajectory as a whole, but also assign stronger learning signals to planning actions that are more important for task success.

Trajectory-level feedback is useful but insufficient for this purpose. A reward indicating whether both objects are placed into the basket can tell us whether the whole rollout succeeds. GRPO-style optimization can further compare several rollouts under the same instruction and assign higher advantages to better trajectories. However, such feedback is still coarse at the action level. A successful rollout contains both planning actions, such as grasping and releasing, and ordinary execution actions, such as small pose adjustments and transitions. If all actions receive the same trajectory-level advantage, the optimization may not sufficiently distinguish which actions have larger influence on the final outcome.

Based on this motivation, PAPO-VLA explicitly models planning actions during optimization. We first identify planning actions using two cues. The first cue is action variation, because planning actions often appear around changes in execution intention. The second cue is trajectory outcome, because a large action change in a failed trajectory may simply reflect unstable control or an unsuccessful attempt. After identifying planning actions, we estimate their importance through causal sufficiency and causal necessity. Sufficiency asks whether preserving a planning action better supports subsequent successful execution. Necessity asks whether perturbing the planning action makes the task outcome



(a) Task 1 (b) Task 2

Figure 4: Qualitative Analysis for PAPO-VLA.

worse. These two criteria allow PAPO-VLA to emphasize planning actions that are more important for task success, while still preserving the trajectory-level learning signal from GRPO.

E.3 Qualitative Analysis

Figure 4 provides an illustration of how PAPO-VLA assigns planning-action importance. We show two representative manipulation tasks: "putting both the alphabet soup and tomato sauce into the basket", and "putting both the cream cheese box and butter into the basket". These tasks require the policy to complete several ordered manipulation steps, such as locating the target objects, approaching them, adjusting the end-effector, grasping, moving, and placing them into the target container.

The visualization shows that the identified planning actions usually appear around important execution moments in the trajectory. For example, actions around approaching the target object, forming a stable grasp, moving the object toward the basket, or releasing it into the basket tend to receive higher planning-action importance. These actions correspond to task-oriented decision steps in the trajectory, because they have stronger influence on whether the subsequent execution can remain on a successful path. In contrast, many other actions mainly maintain motion continuity, make small pose adjustments, or connect one stage of execution to the next. These actions are more related to the executor role of the policy.

E.4 Broader Impacts and Limitations

In this section, we provide a brief illustration of broader impacts and limitations of this work.

Broader Impacts. This work studies policy optimization for Vision-Language-Action models and proposes PAPO-VLA, which identifies planning actions in dense execution trajectories and measures their importance during VLA post-training. By encouraging the policy to place stronger emphasis on planning actions that are more important for task success, this work may help improve the reliability of language-guided robotic manipulation. Beyond the specific benchmarks evaluated in this paper, the proposed formulation also provides a possible direction for understanding and optimizing sequential robotic actions from the perspective of planner and executor roles. It may inspire future research on planning-aware policy optimization, causal analysis for embodied agents, and more effective post-training methods for VLA models. At the same time, more capable robotic manipulation systems should be deployed with appropriate safety constraints, human supervision, and environment-specific validation, especially when moving from simulation to real-world settings.

Limitations. Our main benchmark evaluations are conducted in simulation, supplemented by a controlled real-robot evaluation on six tabletop tasks. We have not yet evaluated PAPO-VLA on larger-scale VLA backbones or broader real-world deployment settings. Future work may apply PAPO-VLA to stronger and larger VLA backbones to further assess its scalability and robustness.

Table 4: Selected RoboTwin2.0 tasks grouped by execution horizon.

Horizon Group	Task	Steps
Short	`lift_pot`	112
	`beat_block_hammer`	113
	`pick_dual_bottles`	127
	`place_phone_stand`	130
Medium	`move_can_pot`	151
	`place_a2b_left`	155
	`place_empty_cup`	174
	`handover_mic`	223
Long	`handover_block`	283
	`stack_bowls_two`	313
Extra-Long	`blocks_rank_rgb`	466
	`put_bottles_dustbin`	637

Planning-action identification may also miss important actions with relatively small changes when larger transitions dominate the top-k candidates. The predefined candidate budget may further limit coverage in longer trajectories with more decision points. Increasing k may improve candidate coverage, but requires additional causal evaluations.

In our experiments, all tasks use the same binary task-success reward $r(\tau) \in \{0, 1\}$ and a shared η, so reward-scale differences do not arise across tasks. For dense rewards with different numerical ranges, a practical extension is to use the normalized reward $\tilde{r}(\tau) = (r(\tau) - r_{\min})/(r_{\max} - r_{\min})$ when computing causal sufficiency and necessity, reducing their dependence on the raw reward scale.

E.5 Use of Large Language Models

Large language models (LLMs) were used solely as a general-purpose tool for grammar and language polishing. They were not involved in research ideation, methodology, analysis, or substantive writing. The authors take full responsibility for all contents of the paper.

F Implementation Details

We implement PAPO-VLA based on open-source VLA codebases. For LIBERO and RoboTwin2.0, we use OpenVLA-7B as the backbone, retaining its original policy architecture and action prediction head. We adopt rank-32 LoRA for RL post-training, without additional pretraining or supervised fine-tuning beyond the released initialization checkpoints. Specifically, the policy is initialized from the official OpenVLA checkpoint for LIBERO and the official OpenVLA-OFT checkpoint for RoboTwin2.0. PAPO-VLA and the corresponding GRPO baseline share the same backbone, initialization, and common training settings. PAPO-VLA additionally performs intervention-based rollouts to estimate planning-action importance and incorporates this signal into advantage estimation.

For the core components of PAPO-VLA, directly computing causal sufficiency and necessity for all actions in all sampled trajectories would introduce excessive rollout cost. Therefore, for each group of G trajectories with rewards $\{r^i\}_{i=1}^G$, we first select the Top S trajectories according to their trajectory-level rewards:

$$\mathcal{I}_S = \mathrm{TopS}(\{r^i\}_{i=1}^G, S). \tag{13}$$

For trajectories not selected into $\mathcal{I}_S$, we set their additional planning-action importance to zero. These trajectories still participate in GRPO through their trajectory-level advantages. This design only avoids extra causal importance estimation on trajectories whose executions are less likely to provide reliable positive planning-action signals. In our experiments, we set $S = 2$.

Within each selected trajectory, we identify planning actions according to action variation and trajectory outcome, and obtain the planning-action mask $m_{i,t}^{\mathrm{plan}}$. For non-planning actions, we directly set their planning-action importance to zero. The number of selected planning actions is set

according to the task horizon:

$$k = \begin{cases} 2, & \text{short horizon tasks,} \\ 4, & \text{medium horizon tasks,} \\ 6, & \text{long horizon tasks.} \end{cases} \tag{14}$$

This setting keeps the number of additional causal evaluations bounded while allowing longer tasks to include more planning actions. For LIBERO, we use $k = 2$ for Spatial and Object, $k = 4$ for Goal, and $k = 6$ for Long. For RoboTwin2.0, the short-, medium-, and long/extra-long-horizon groups use $k = 2$, $k = 4$, and $k = 6$, respectively. The value of k is used only during training and is not required during inference.

For the selected planning actions, the expectation terms in causal sufficiency and necessity are estimated by Monte Carlo sampling averages. For each selected planning action, we apply a single-axis Rademacher perturbation. Specifically, we uniformly sample one continuous control dimension j and a direction $s \in \{-1, +1\}$, and construct the perturbed action as $\bar{a}_{t,j} = \mathrm{clip}\left(a_{t,j} + s\rho(a_{\max,j} - a_{\min,j}), a_{\min,j}, a_{\max,j}\right)$. The other action components remain unchanged, and the per-dimension bounds are obtained from the action statistics associated with the corresponding checkpoint. The perturbed action is executed in the simulator, after which the policy continues generating subsequent actions in closed loop from the resulting observation, rather than replaying the original future actions. We select $\rho = 0.01$ from $\{0.0025, 0.005, 0.01, 0.02, 0.025\}$ to balance observable changes in execution against excessive disruption of the overall trajectory. For causal sufficiency, the implementation estimates the expectation terms in **Eq. 8** as

$$\widehat{C}^{\text{suff}}_{i,t} = m^{\text{plan}}_{i,t} \left[\frac{1}{M} \sum_{m=1}^{M} r(\tau^{i,m}_{\leq t}) - \frac{1}{M} \sum_{m=1}^{M} r(\bar{\tau}^{i,m}_{\leq t}) \right]_+, \tag{15}$$

where $\tau^{i,m}_{\leq t} \sim \pi(\cdot \mid \mathcal{A}^i_{\leq t})$ denotes a rollout that preserves the original action prefix, and $\bar{\tau}^{i,m}_{\leq t} \sim \pi(\cdot \mid \bar{\mathcal{A}}^i_{\leq t})$ denotes a rollout that replaces a^i_t with the perturbed action $\bar{a}^i_t$ in the prefix. For causal necessity, the implementation estimates the expectation terms in **Eq. 9** as

$$\widehat{C}^{\text{nec}}_{i,t} = m^{\text{plan}}_{i,t} \left[\frac{1}{M} \sum_{m=1}^{M} r(\tau^{i,m}_{\mathcal{A}}) - \frac{1}{M} \sum_{m=1}^{M} r(\tau^{i,m}_{\bar{\mathcal{A}}_t}) \right]_+, \tag{16}$$

where $\tau^{i,m}_{\mathcal{A}} \sim \pi(\cdot \mid \mathcal{A}^i)$ denotes a rollout under the original action sequence, and $\tau^{i,m}_{\bar{\mathcal{A}}_t} \sim \pi(\cdot \mid \bar{\mathcal{A}}^i_t)$ denotes a rollout where the planning action a^i_t is replaced by $\bar{a}^i_t$. In our experiments, we set $M = 1$.

To further reduce the rollout cost, we reuse the original trajectory reward as one side of the comparison whenever possible. Since the original trajectory already provides one sample under the preserved-action condition in sufficiency estimation and under the original-sequence condition in necessity estimation, we approximate

$$\frac{1}{M} \sum_{m=1}^{M} r(\tau^{i,m}_{\leq t}) \approx r(\tau^i), \quad \frac{1}{M} \sum_{m=1}^{M} r(\tau^{i,m}_{\mathcal{A}}) \approx r(\tau^i). \tag{17}$$

Thus, only the comparison branches $\bar{\mathcal{A}}^i_{\leq t}$ and $\bar{\mathcal{A}}^i_t$ require additional rollout sampling. This approximation keeps the additional cost controlled while preserving the causal comparison between preserving and perturbing a planning action.

Definition 1 characterizes causal sufficiency and necessity through conditional interventional probabilities. In practice, **Eq.8** and **Eq.9** use positive differences in expected rollout rewards as surrogate measures. Since we adopt binary task-success rewards, each expected reward corresponds to a success probability under the respective rollout condition, although the resulting differences are not exact estimates of the conditional probabilities in **Definition 1**.

During RL post-training, trajectories are generated online through closed-loop interaction between the policy and the simulator under each task condition, and each trajectory receives a binary task-success reward $r(\tau) \in \{0, 1\}$. The group size is set to $G = 8$. We use a learning rate of 5×10^{-6}, training batch size of 64, mini-batch size of 128, rollout sampling temperature of 1.6, and PPO-style clipping thresholds $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$. The number of action chunks is set to 25 for RoboTwin2.0. The maximum environment interaction step is set to 512 for LIBERO and to 200, 400, or 800 for



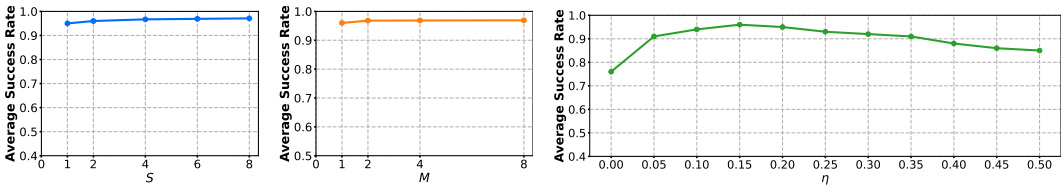
Figure 5: More Visualization of OpenVLA and Ours.

RoboTwin2.0 according to the task horizon. For RoboTwin2.0, we select 12 tasks covering different manipulation behaviors and execution lengths. Based on the required execution steps, these tasks are categorized into four groups: short, medium, long, and extra-long horizons. The selected tasks span 112–637 steps, with the detailed task classification provided in **Table 4**. All experiments are conducted on H100 GPU clusters.

G More Experiments and results

G.1 More Visualization and Case Study

Figure 5 provides additional qualitative comparisons between OpenVLA and our method on four manipulation tasks: "picking up the black bowl on the ramekin and placing it on the plate", "putting the bowl on the stove", "opening the top drawer and putting the bowl inside", and "putting both moka pots on the stove". These tasks require the policy to complete several ordered actions, such as



(a) S v.s. LIBERO Average Success Rate (b) M v.s. LIBERO Average Success Rate (c) η v.s. LIBERO Average Success Rate

Figure 6: Parameter Sensitivity.

Table 5: Real-robot evaluation results on six manipulation tasks.

Task	Standard GRPO	PAPO-VLA
Place an apple into a basket	0.80	0.80
Pour beans into a cup	0.30	0.50
Turn a mug upright	0.60	0.80
Move a toy block onto a tray	0.40	0.70
Knock over a bottle	0.50	0.60
Cover a bowl with a cloth	0.40	0.70
Average	0.500	**0.683**

approaching the target object, aligning the gripper, grasping or moving the object, and placing it at the target location.

Compared with OpenVLA, PAPO-VLA produces more complete execution trajectories in these cases. In particular, our method can better maintain task progress across multiple stages and reduce incomplete executions caused by missing important intermediate actions. By incorporating planning-action importance into policy optimization, PAPO-VLA can provide more focused guidance for these actions, leading to more reliable task success.

G.2 Real-Robot Evaluation

To further evaluate PAPO-VLA under physical execution conditions, we conduct real-robot experiments on six tabletop manipulation tasks using a Franka Emika Panda 7-DoF robotic arm equipped with a Robotiq gripper. These tasks include placing an apple into a basket, pouring beans into a cup, turning a mug upright, moving a toy block onto a tray, knocking over a bottle, and covering a bowl with a cloth. They cover different manipulation behaviors, including grasping and placement, object reorientation, pouring, contact-based interaction, and multi-stage manipulation. Visual observations are captured using a fixed third-person RGB camera. At each policy step, the model receives the current RGB observation and task instruction, and predicts a 7-dimensional relative end-effector action consisting of three translational, three rotational, and one gripper component. The robot operates at 5 Hz through a Cartesian control interface. For policy adaptation, two human operators collect demonstrations using a 3Dconnexion SpaceMouse to control the robot's end-effector motion and gripper. Each operator collects 20 demonstrations per task, resulting in 40 demonstrations per task and 240 trajectories across the six tasks. PAPO-VLA and standard GRPO use the same real-robot demonstrations, supervised adaptation procedure, and adaptation configuration. Both methods also share the same robot-control system, task settings, and success criteria, ensuring a consistent comparison.

During evaluation, manipulated objects and target objects are initialized within predefined placement regions, with object positions and relevant orientations varied across trials. Both methods are evaluated under the same initial-state ranges and physical configurations. Each method is evaluated over 10 closed-loop trials per task, resulting in 120 evaluation trials across the six tasks. A trial is considered successful only when the policy completes the entire task objective within the predefined time limit without human intervention. Partial completion is counted as failure, while trials interrupted by external system errors unrelated to policy execution, such as robot communication or data-recording failures, are treated as invalid and repeated. **Table 5** presents the real-robot evaluation results,

Table 6: Results of different GRPO variant.

Method	Spatial	Object	Goal	Long	Avg.
Dr.GRPO	0.88	0.92	0.85	0.73	0.85
Dr.GRPO + PAPO	0.92	0.96	0.94	0.89	0.93

Table 7: The effect of Planning Action Identification.

Action	**LIBERO Average Success Rate**	**Relative Cost (GRPO: 1.00×)**
Candidate Action Selection	0.96	1.58×
Full Action Sequence	0.97	33.78×

providing additional empirical evidence for the effectiveness of PAPO-VLA under real-world robotic manipulation conditions.

G.3 Different GRPO Variant

We further evaluate the compatibility of PAPO-VLA with Dr.GRPO, which removes group-wise reward standard-deviation normalization and trajectory-length-dependent loss normalization. We incorporate planning-action importance into the Dr.GRPO advantage while retaining the same backbone, initialization, and optimization settings. As shown in **Table 6**, incorporating PAPO improves the average success rate.

G.4 Effect of Planning Action Identification

Table 7 analyzes the effect of planning action identification. We compare our selective strategy with a variant that uses the full action sequence for planning-action importance estimation. The full-sequence variant treats all action positions as possible planning actions, which weakens the focus on task-oriented decision steps and also introduces substantially higher computation, since causal sufficiency and necessity need to be estimated for many more action positions. In contrast, our planning action identification keeps only a small set of actions that are more likely to reflect changes in execution intention, according to action variation and trajectory outcome. This makes the importance estimation more focused and efficient.

The results show that planning action identification achieves a better balance between performance and computation. This indicates that estimating causal importance on a selected set of planning actions achieves comparable performance with substantially lower computational cost. It also supports our motivation that VLA optimization should not treat all actions equally, but should place stronger emphasis on planning actions that have larger influence on task success.

G.5 Parameter Sensitivity

Figure 6 and **Table 8** show the parameter sensitivity results. We study four important hyperparameters: the number of selected trajectories S, the number of Monte Carlo samples M, the planning-action importance coefficient η, and the number of selected planning actions k.

For S in **Figure 6a**, the performance improves when a small number of high-quality trajectories are used for planning-action importance estimation, and the best trade-off is obtained around $S = 2$. Increasing S further does not bring a clear performance gain, while it directly increases the number of trajectories that require causal estimation. Thus, we set $S = 2$ as our final configuration

For M in **Figure 6b**, increasing M can make the Monte Carlo estimation better in principle, but the empirical improvement is limited. Since the rollout cost grows almost linearly with M, using more samples would significantly increase training cost without clear performance improvement. Therefore, we set $M = 1$ in the main experiments.

For η in **Figure 6c**, the performance first improves as the planning-action importance term becomes effective, and then decreases when the causal term becomes too large. A small η cannot sufficiently reflect planning-action importance, while an overly large η may weaken the role of the original

Table 8: Parameter sensitivity of the number of selected planning actions k.

	LIBERO				RoboTwin2.0		
k	Spatial	Object	Goal	Long	Short (%)	Medium (%)	Long/Extra-long (%)
2	0.93	0.98	0.96	0.91	58.6	68.7	61.9
4	0.93	0.98	0.98	0.92	58.8	69.5	62.3
6	0.94	0.98	0.98	0.94	58.9	69.7	63.8
8	0.94	0.99	0.98	0.94	58.9	69.8	63.9

Table 9: Trade-off Performance.

Method	**Avg Step Time**	**Peak GPU Mem**	**Relative Time**	**Relative Mem**
GRPO	0.425 s/step	19.91 GiB	1.00×	1.00×
Ours	0.672 s/step	20.90 GiB	1.58×	1.05×

trajectory-level advantage. The best result is obtained at $\eta = 0.15$, which is used as the default setting in our experiments.

For k in **Table 8**, the performance generally improves as k increases, with more noticeable gains on longer-horizon tasks. However, increasing k further brings limited performance improvement, while introducing additional causal evaluation costs. Therefore, we set k according to the task horizon, as specified in **Appendix F**.

G.6 Trade-off Performance

Table 9 reports the trade-off between performance and computational cost. Compared with standard GRPO, PAPO-VLA introduces additional computation because it estimates causal sufficiency and necessity for selected planning actions. This mainly increases rollout time, since extra environment interactions are needed to evaluate the effect of preserving or perturbing these plannning actions. However, the memory overhead remains small because PAPO-VLA does not introduce a new policy backbone, value model, or separate reward model.

Overall, the results indicate that PAPO-VLA achieves a reasonable trade-off between effectiveness and efficiency. By focusing the additional estimation on a small set of planning actions, PAPO-VLA can incorporate planning actions into policy optimization without applying extra causal evaluation to the full action sequence.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 main claim of this paper and the scope of the method are stated in the abstract and introduction, and are supported by the experiments.

   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: We discuss limitations in Appendix E.4.

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: [N/A]

   Justification: We do not present formal theoretical theorems or proofs. We use causal sufficiency and necessity as methodological definitions for action-level contribution estimation.

   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: We provide experimental settings, benchmark descriptions, implementation details, pseudo-code, and additional analyses in Section 5 and Appendix G.

   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: We will make our code available for public soon.

   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: We have provided experimental details and results in Section 5 and Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report results and details in Section 5 and Appendices.

   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: We have specified these in Section 5 and Appendix F.

   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: This work conforms, in every respect, with the NeurIPS Code of Ethics, e.g., anonymity.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We provide both potential positive and negative societal impacts in Appendices.

    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: [Yes]

    Justification: We will provide the usage guidelines or restrictions to access the model or implementing safety filters when make our model public.

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: We have cite all used data and provided the licenses properly.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We have cite all used data properly in Appendices.

    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 involve crowdsourcing, user studies, or human-subject research.

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 involve crowdsourcing, user studies, or human-subject research.

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: We only use LLM for editing or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research.

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.