VLA-ROLLING: FEWER DENOISING STEPS, BETTER CONTROL VIA ADAPTIVE ACTION REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching vision-language-action (VLA) models generate an action chunk by iteratively denoising Gaussian noise, but typically execute only a short prefix before replanning with updated observations. Under this receding-horizon paradigm, the unexecuted remainder—the *action tail*—is discarded, and the next chunk is regenerated from fresh noise. We show that this discarded tail retains *transferable denoising progress*: action-tail initialization can match or outperform a native 10-step Gaussian baseline using fewer denoising steps. However, the optimal denoising budget varies substantially across samples, indicating that tail reuse should be adapted to the updated environment. Based on this observation, we introduce **VLA-Rolling**, a lightweight inference framework that reuses action tails through three stages: action-tail inheritance, per-action confidence estimation, and confidence-guided refinement through renoising and denoising budget allocation, while the underlying VLA remains frozen. Across $\pi_{0.5}$, SmolVLA, X-VLA, and MolmoAct2 on LIBERO, MetaWorld, LIBERO-Plus, and RoboTwin 2.0, VLA-Rolling reduces action-chunk inference cost in all 16 model–benchmark pairs, achieving up to a $1.48\times$ speedup, while improving success rate in 14 pairs by up to 15.08 percentage points. These results show that action tails are not merely discarded predictions, but reusable intermediate states that enable more efficient and accurate VLA inference. These results demonstrate that the action tail can support the generation of better actions with fewer denoising steps.

1 INTRODUCTION

Flow-matching vision-language-action (VLA) models generate future robot actions from visual observations and language instructions (Black et al., 2024; Intelligence et al., 2025; Shukor et al., 2025; Zheng et al., 2026; Lee et al., 2025; Bjorck et al., 2025; Cai et al., 2026). Their architectures differ, but they share a flow-matching action generator that iteratively transforms Gaussian noise into a sequence of future actions, called **action chunk** (Jing et al., 2025; Liang et al., 2026; Xu et al., 2026; Wang et al., 2026a; Lu et al., 2026). In models such as $\pi_{0.5}$ (Intelligence et al., 2025) and SmolVLA (Shukor et al., 2025), this generator is implemented as a dedicated Transformer module called the **action expert**, conditioned on features from a vision-language model (VLM). Thanks to this , the robot can plan and execute several coherent steps without calling the model after every action. Yet the whole chunk is predicted from one observation, so environmental changes can make later actions inaccurate. To limit this error accumulation, **receding-horizon execution** runs only a short prefix, then observes and replans (Zhao et al., 2023; Jing et al., 2025; Liang et al., 2026). The price of this feedback is a fresh start: the unexecuted remainder, or **action tail** (Huang et al., 2026), is discarded, and the next chunk starts again from Gaussian noise.

A fresh start may waste completed computation. The action tail has already undergone denoising and overlaps in time with the actions needed in the next inference round, so it may be a better starting point than fresh Gaussian noise. Yet the tail was generated from an earlier observation and may have become stale. It is neither a ready-to-execute plan nor fresh noise: its value as a starting point depends on how much the environment has changed. This tension raises a compelling question:

*Can we exploit the action tail to generate new actions **faster** without sacrificing action **accuracy**?*

We probe this possibility on $\pi_{0.5}$ and SmolVLA (Section 2). On LIBERO and MetaWorld, tail initialization matches or improves the action error of a 10-step Gaussian baseline in four denoising

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

steps for $\pi_{0.5}$ and six for SmolVLA (Figure 3). Yet no fixed short budget suits every sample: the sample-wise optimal step counts span the full range from 1 to 10 for both models (Figure 2). The question is therefore not simply whether to reuse the tail, but how much to trust it under each new observation.

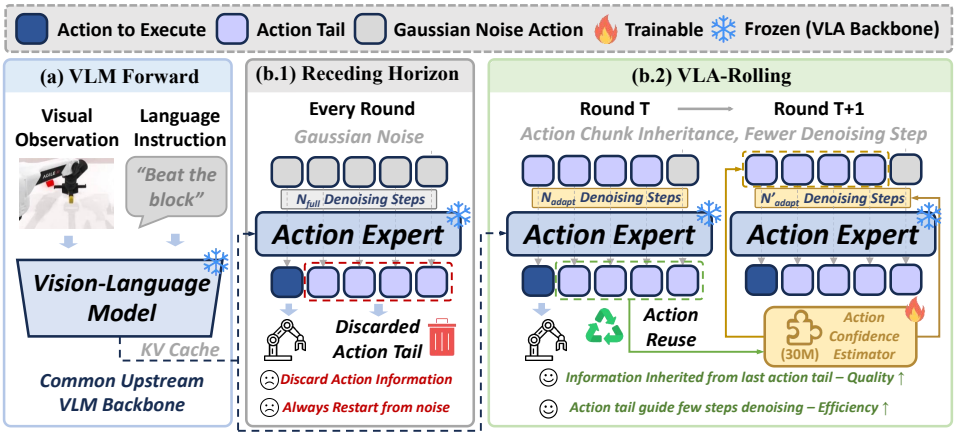


Figure 1: **VLA-Rolling reuses rather than discards action tails.** Confidence controls both how strongly inherited actions are renoised and how many denoising steps refine the next chunk.

To adapt reuse to each new observation, we introduce **VLA-Rolling**, a three-stage inference framework (Figure 1). First, **action-tail inheritance** aligns unexecuted actions with the next planning window, using completed denoising work as the starting point for a new prediction. Second, a lightweight **action confidence estimator** predicts the reuse value of each inherited action from the new observation and state change. We train it offline on temporally adjacent demonstrations: smaller error between an inherited action and its aligned demonstrated action yields a higher confidence target, while the loss penalizes overestimated confidence more heavily. Third, **confidence-guided refinement** injects more noise into low-confidence actions and assigns more denoising steps when the chunk contains an unreliable action. The underlying VLA remains frozen.

To summarize, our main contributions are:

(1) *Action-Tail Reuse Potential* (Section 2): We reveal that discarded tails retain transferable denoising progress: four steps for $\pi_{0.5}$ and six for SmolVLA match a 10-step Gaussian baseline. Yet the sample-wise optimal budgets span 1–10 steps, exposing both the opportunity for reuse and the need to adapt it to each new observation.

(2) *Confidence-Guided Rolling Reuse* (Section 3): We propose VLA-Rolling, a lightweight inference framework that learns per-action reuse confidence from temporally adjacent demonstrations. This confidence jointly controls how much of each inherited action to retain and how many denoising steps to allocate, preserving useful motion while correcting stale predictions without retraining the base VLA.

(3) *Extensive Experiments and Analysis* (Section 4): On $\pi_{0.5}$, SmolVLA, X-VLA, and MolmoAct2 across LIBERO, MetaWorld, LIBERO-Plus, and RoboTwin 2.0, VLA-Rolling reduces action-chunk inference time in all 16 pairs, achieving up to a $1.48\times$ speedup (32.52% lower cost), and improves success in 14 pairs by up to 15.08 percentage points. Ablations expose the risk of naive reuse, while real-world target changes trigger additional refinement.

2 Probing the Feasibility of Action Tail Reuse

Flow-matching VLA jointly predicts an H-step action chunk from a single environment state $\mathbf{s}_t$. However, during the execution of the action chunk, the latest observations from the environment are not used to update the action, causing all subsequent actions to be based on the state observed before execution began. Errors in early actions cause a deviation between the actual state of the robot and

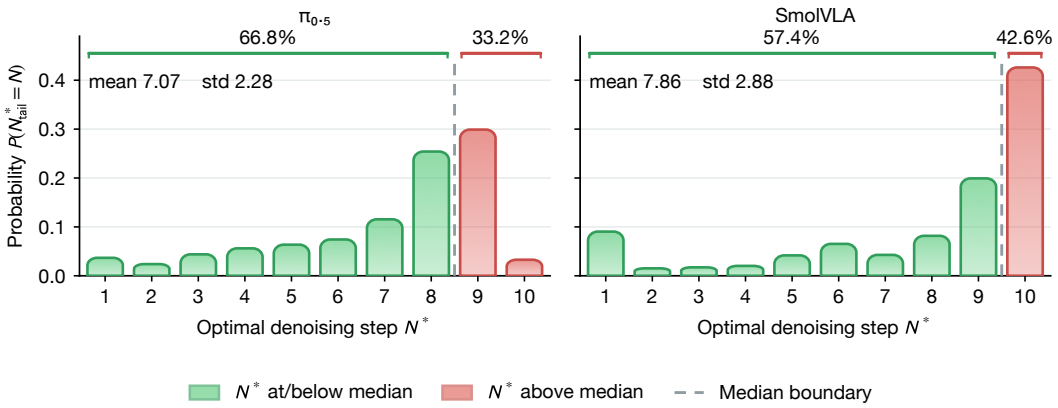


Figure 2: **Action tails require sample-dependent refinement.** Each bar shows the fraction of samples with a given optimal denoising step N^*. Green denotes N^* at or below the model-specific median (8 for $\pi_{0.5}$, 9 for SmolVLA), while red denotes N^* above it. Dashed lines mark the median boundaries; brackets report the fraction of samples in each region. Means and standard deviations appear inside the panels.

the environment and the state expected by the remaining predicted actions, and this state mismatch propagates to subsequent actions, gradually accumulating throughout the execution scope.

Receding-horizon execution mitigates such error accumulation by periodically closing the perception–action loop. Instead of executing all H predicted actions, the robot only the first K actions (where $K < H$), then obtains the updated observations and generates a new action block. This execution paradigm consumes more inference iterations in exchange for the ability to use new observations to update actions more frequently.

This execution strategy also leaves part of every generated action chunk unused. After executing the first K actions, the remaining $L = H - K$ actions are

$$\mathbf{A}_t^{\text{tail}} = \mathbf{A}_t^{K:H}, \tag{1}$$

which we refer to as the **action tail**. Under conventional receding-horizon execution, this tail is discarded once the observation is refreshed. However, our statistics show that under the execution horizon of common VLA models, the environment does not change significantly. Observations and proprioceptions have high similarity in nearly 98% of adjacent environment steps. Detail statistical data is shown in the Appendix. D.5.

Diagnostic Evaluation Setup. We conducted diagnostic experiments on $\pi_{0.5}$ and SmolVLA to analyze the properties of the action tail after updating observations. For two time steps $t - K$ and t separated by K control steps in the same demonstration trajectory, we first generate an action chunk of length H using the frozen VLA in the old state $\mathbf{s}_{t-K}$. After assuming the first K actions have been executed, the remaining $L = H - K$ actions constitute the action tail. This tail is temporally aligned with the demonstration action starting at time t. Subsequently, in the updated state $\mathbf{s}_t$, we initialize with this action tail and fresh Gaussian noise respectively, and compare them using the same target action $\mathbf{A}_t^*$.

Diagnostic Evaluation Result. We first investigate whether it's possible to generate the target action with fewer denoising steps by starting from the sampled action tail. Specifically, we perform $N \in \{2, 4, 6, 8\}$ steps of denoising on each inherited action tail, and compare the final error E_{tail}^N with the error of the complete Gaussian noise denoising result $E_{\text{noise}}^{N_{full}}$.

Figure 3 shows that an inherited tail can surpass the 10-step Gaussian baseline in fewer denoising steps: four for $\pi_{0.5}$ and six for SmolVLA on both benchmarks. However, more steps do not always improve the prediction. For $\pi_{0.5}$ on LIBERO, the error ratio falls from 1.073 at six steps to 1.037 at eight, indicating a larger action error after further refinement. This raises a sample-level question: *How many denoising steps does each inherited tail need?*

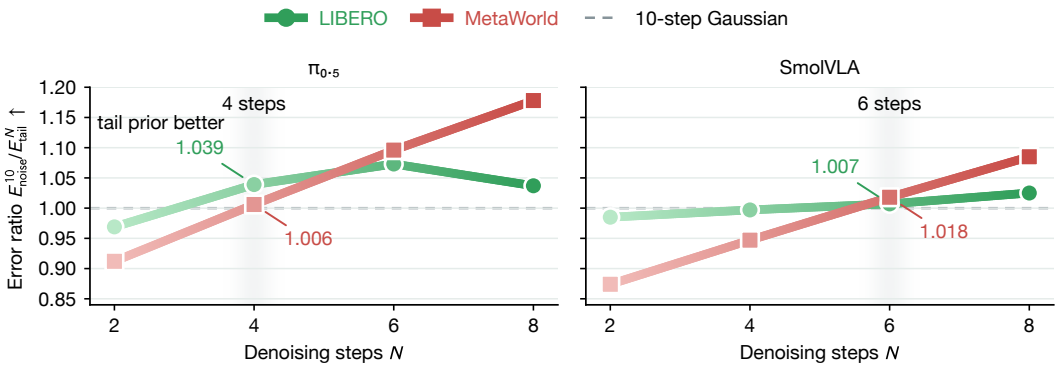


Figure 3: **Action tails recover a 10-step Gaussian prediction in fewer steps.** Tail initialization crosses the Gaussian baseline in four steps for $\pi_{0.5}$ and six for SmolVLA on both benchmarks. Subtle shaded bands mark these earliest shared operating points.

We define the sample-wise optimal denoising step as

$$N^* = \arg\min_N E^N_{\text{tail}} \tag{2}$$

where $N \in \{1, \ldots, N_{full}\}$. We then analyze the distribution of N^* for both models.

As shown in Figure 2, N^* spans every step from 1 to 10, with standard deviations of 2.28 for $\pi_{0.5}$ and 2.88 for SmolVLA. Even beyond the respective medians of 8 and 9 steps, 33.2% and 42.6% of samples require further refinement. This variation motivates selecting the denoising budget separately for each sample.

Together, these diagnostics motivate a sample-adaptive budget: reuse the tail when fewer denoising steps suffice, while allowing further refinement when needed.

3 VLA-Rolling: Adaptive Action Tail Reuse

VLA-Rolling attempts to utilize widely existing but underutilized information: the **action tail**. VLA-Rolling is a inference paradigm that uses the action tail of each round as prior knowledge for the next inference round, as shown in Figure 4. VLA-Rolling are designed into three phases: **(1) Action-tail inheritance stage** (Section 3.1) is used to retain the action and environment state obtained by the previous model after denoising, which serves as the basis for building the starting point of the current denoising stage; **(2) Action tail estimation stage** (Section 3.2) uses the trained action confidence estimator to evaluate the retention value of the action based on the environment state and its changes; **(3) Confidence-guided refinement stage** (Section 3.3) generates the starting point of the action by adding noise and assigns it the corresponding denoising cost based on the obtained action tail and confidence.

3.1 Action Tail Inheritance

During the action inheritance phase, We obtain the preserved action tail from previos round, temporally aligne it with action at time t.

$$\mathbf{A}_{t-K} = \pi_\theta(\mathbf{o}_{t-K}, \mathbf{q}_{t-K}, \ell), \qquad \mathbf{A}^{tail}_{t-K} = \mathbf{A}^{K:H}_{t-K} \in \mathbb{R}^{L \times d_a}, \qquad L = H - K, \tag{3}$$

Meanwhile, for the following phases, we need to use the environment state saved in the previous round, and after obtaining the environment state in this round, calculate the corresponding environmental change:

$$\Delta \mathbf{o_t} = \mathbf{o_t} - \mathbf{o_{t-k}} \qquad \Delta \mathbf{q_t} = \mathbf{q_t} - \mathbf{q_{t-k}} \tag{4}$$

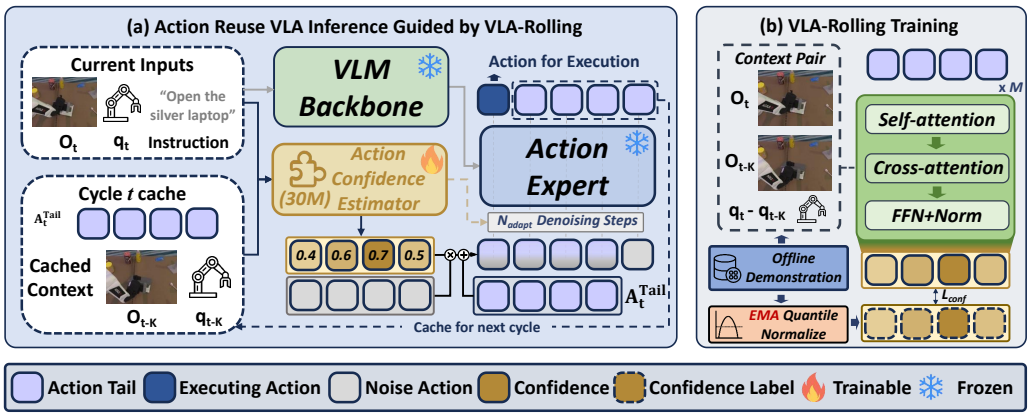


Figure 4: **Overview of VLA-Rolling.** VLA-Rolling reuses the unexecuted action tail from the previous inference round by estimating its confidence, which jointly determines tail preservation and the adaptive denoising budget while keeping the original VLA frozen. **(a) VLA-Rolling inference.** The Action Confidence Estimator predicts a confidence which controls its interpolation with Gaussian noise. The resulting initialization is refined using N_{adapt} denoising steps. The executable prefix is sent to the robot, while the remaining tail rolls into the next round. **(b) VLA-Rolling training.** Confidence labels are constructed from the quality of inherited actions and normalized with EMA-based quantile statistics.

where observation $\mathbf{o}_t$ and proprioceptive state $\mathbf{q}_t$ denote the current environmental state, $\Delta\mathbf{o}_t$ and $\Delta\mathbf{q}_t$ denote the environmental change.

3.2 Action Confidence Estimation

To estimate the number of denoising steps required for the current action, we propose the **Action Confidence Estimator**, a lightweight plugin that predicts the confidence of the action tail based on the current environmental changes and state. We formalize action confidence estimator as f_ϕ:

$$c_\phi = f_\phi\left(\mathbf{A}^{\text{tail}}_{t-K} \,\middle|\, \mathbf{o}_t, \Delta\mathbf{o}_t, \mathbf{q}_t, \Delta\mathbf{q}_t\right), \tag{5}$$

The output $c \in \mathbb{R}^{L\times 1}$, represents the estimated reliability of retaining the inherited action at corresponding position.

To supervise action confidence estimator to learn representing the reusability of actions under changing environments, we choose the error between the action tail and the target action E_t as the representation of behavior in the new environment. This is because the closer the action tail was to the target in the previous denoising process, the easier it is to refine, and therefore it should receive a higher confidence. We then perform a logarithmic transformation on E_t to obtain the risk r_t. Because the action tail is already quite close to the target action, this transformation makes subtle differences in the action more discernible.

$$E_t = \left\|\mathbf{A}^{tail}_{t-K} - \mathbf{A}^{*}_{t}\right\|_2^2, \qquad r_t = \log(E_t), \tag{6}$$

Given the wide distribution of action errors, we use log transformation to compress larger error ranges while preserving discriminative power in smaller error regions. r_t measures the absolute deviation of the inherited action relative to the aligned demonstration action.

Meanwhile, considering that action confidence estimator needs to be adapted to different models and environments, we perform quantile normalization on risk to unify the error scale:

$$c^*_t = 1 - \text{clip}\left(\frac{r_t - q_{0.1}}{q_{0.9} - q_{0.1}}, 0, 1\right). \tag{7}$$

Action confidence estimator is trained by predicting the difference between confidence and risk confidence target. Our training objective is formulated as:

$$\mathcal{L}_{\text{conf}} = \mathbb{E}_{\mathbf{A}_t^{tail}, o_{t-K}, o_t, q_{t-K}, q_t} \left[\| w_i \left(c_\phi - c^* \right) \|_1 \right], \qquad w_i = \begin{cases} w_+, & c_{\phi,i} > c^*_{t,i}, \\ w_-, & c_{\phi,i} \leq c^*_{t,i}, \end{cases} \tag{8}$$

where $w_+ > w_-$ assigns a larger penalty to overestimating the confidence of an action tail.

3.3 Confidence-Guided Refinement

To optimize the inference process through preservable prior information in the action tail, we propose the **Rolling Action Reuse** inference paradigm. The trained action confidence estimator performs a forward operation before each subsequent action generation to obtain the confidence of the tail of the current inherited action. This confidence then guides two aspects of action generation: (1) Renoising determines how much Gaussian noise should be injected as annealing before action generation to reduce the likelihood of actions going out of distribution; (2) Adaptive-Step Denoising determines the denoising budget allocated for action generation.

The prefix of the action start point is determined by renoising. We will use the predicted confidence as the noise linear interpolation coefficient for the action tail at different positions:

$$\mathbf{A}_{t,i}^{\text{init}} = \begin{cases} c_{\phi,i} \mathbf{A}_{t-K,i}^{tail} + (1 - c_{\phi,i}) \boldsymbol{\epsilon}, & 0 \leq i < L, \\ \boldsymbol{\epsilon}_i, & L \leq i < H. \end{cases} \tag{9}$$

Despite confidence-dependent annealing of the inherited action tail, the positions without an inherited counterpart are supplied with Gaussian initialization. High-confidence positions retain more temporal information and receive less noise. Low-confidence positions receive stronger renoising, approaching the native VLA initialization for greater regeneration under the current context when the inherited action is unreliable.

During the adaptive-step denoising stage, the confidence level serves as the basis for determining the number of denoising steps for the entire action chunk:

$$c_{\min} = \min_{0 \leq i < L} c_{\phi,i}, \qquad N_{\text{adapt}} = \text{clip} \left(\lceil (1 - c_{\min}) N_{full} \rceil, N_{\min}, N_{full} \right), \tag{10}$$

where $N_{\min} \geq 1$ and N_{full} bound the budget, equaling to the default denoising steps of the frozen VLA.

In this paradigm, the action tail of each inference round is not discarded, but rather rolled forward temporally. By the time the action is finally to execute, it has already been continuously refined over multiple rounds of rolling.

4 Experiments

We tested on LIBERO (Liu et al., 2023), MetaWorld (Yu et al., 2020), LIBERO-Plus (Fei et al., 2025) and RoboTwin 2.0 (Chen et al., 2025b), using $\pi_{0.5}$ as the main VLA backbone, and SmolVLA, X-VLA, and MolmoAct2 as cross-comparison targets. Our experiments were organized into four parts: **(1) Overall performance** comparisons showing the model's baseline performance on various benchmarks and its performance after deploying VLA-Rolling; and **(2) Ablation experiments** verifying the impact of different components in VLA-Rolling on inference results.;**(3) Analysis Experiments**; **(4) Real-world experiments** verifying the deployment of VLA-Rolling on real-world VLA models.

4.1 Overall Performance

Evaluation Setup. The main experiment was conducted on three simulation benchmarks: LIBERO, MetaWorld, and LIBERO-Plus. LIBERO contains 40 tasks devided into 4 suites: spatial, object, goal and long. For each task, we let the model perform 100 rollouts. MetaWorld

Table 1: Performance and inference efficiency comparison with VLA-Rolling. **SR** denotes the average success rate across tasks, and **Cost** denotes the inference time per action chunk. **Bold** data indicates an improvement compared to the baseline.

Model	LIBERO		MetaWorld		LIBERO-Plus		RoboTwin	
	SR (%)↑	Cost (ms)↓	SR (%)↑	Cost (ms)↓	SR (%)↑	Cost (ms)↓	SR (%)↑	Cost (ms)↓
$\pi_{0.5}$	96.38	168.09	58.56	222.69	84.73	167.65	66.54	172.92
+VLA-Rolling	**97.83**	**129.60**	**67.88**	**184.21**	**85.42**	**147.40**	**67.26**	**164.27**
Δ (pp/%)	**+1.45**	**-22.90%**	**+9.32**	**-17.28%**	**+0.69**	**-12.08%**	**+0.72**	**-5.00%**
X-VLA	96.03	135.24	57.08	123.31	**66.04**	136.80	**60.68**	131.09
+VLA-Rolling	**96.65**	**112.79**	**61.46**	**90.05**	65.91	**121.93**	60.40	**96.99**
Δ (pp/%)	**+0.62**	**-16.60%**	**+4.38**	**-26.97%**	-0.13	**-10.87%**	-0.28	**-26.01%**
SmolVLA	73.85	127.80	65.84	122.88	20.74	130.90	25.36	148.68
+VLA-Rolling	**79.78**	**89.57**	**70.74**	**102.57**	**35.82**	**120.97**	**31.32**	**137.34**
Δ (pp/%)	**+5.93**	**-29.91%**	**+4.90**	**-16.53%**	**+15.08**	**-7.59%**	**+5.96**	**-7.63%**
MolmoACT2	95.50	291.34	28.56	278.53	78.96	303.18	20.18	387.06
+VLA-Rolling	**95.70**	**196.61**	**30.56**	**236.45**	**79.07**	**207.27**	**22.34**	**273.95**
Δ (pp/%)	**+0.20**	**-32.52%**	**+2.00**	**-15.11%**	**+0.11**	**-31.63%**	**+2.16**	**-29.22%**

contains 50 tasks across 4 difficulty levels, every task is evaluated with 100 rollouts. LIBERO-Plus is a perturbed version of the LIBERO dataset, incorporating common perturbations such as lighting and background variations, resulting in a total of 10,030 tasks. We test our model on every task for 1 episode. RoboTwin 2.0 contains 50 bimanual manipulation, owning corresponding variants with random perturbations to test the model's robustness. We ran 100 rollouts for each task.

Evaluation Results. Table 1 shows that VLA-Rolling provides significant Pareto improvements, maintaining or improving task success rates while reducing inference costs. From the results, we can make two observations: First, the model's inference cost is significantly reduced. The time to infer each action chunk on different models and benchmarks is reduced by 5.00%-32.52%, equivalent to an increase of up to 1.48 times in action chunk throughput. This improvement comes from VLA-Rolling's adaptive denoising step estimation, which provides the VLA model with guidance for fewer denoising steps. Second, the task success rate on each benchmark remains basically unchanged, or even slightly improved. This improvement comes from lower action error at each step of action generation, resulting in less cumulative error and an increase in the number of tasks completed. These results demonstrate that VLA-Rolling provides significant Pareto improvements, maintaining or improving task success rates while reducing inference costs.

4.2 Ablation Study

Table 2 shows the component ablation. We can observe that simply inheriting the action tail reduced the success rate by 4.89%, while merely adjusting the number of denoising steps also reduced it by 4.29%. All the success rate improvements occurred in variants with renoising.

From this, we can draw two conclusions: First, even if the inherited action tail is closer to the target distribution than Gaussian noise, we must consider the characteristic that high-step denoising of the action tail impairs action quality (as mentioned in Section 2). Therefore, directly inheriting the action tail and performing full-step denoising will only reduce the success rate. Second, the renoising process can significantly improve model performance. This is because the action itself sometimes lies outside the training distribution, making it difficult to correctly refine it using only the VLA's capabilities; adding a small amount of noise can reduce this possibility.

4.3 Analysis Experiments

Environment Shift Study. We explored an action confidence estimator model trained using default settings, based on $\pi_{0.5}$, and inputting environmental states at different execution time domains K to simulate time shifts. The larger K is, the greater the environmental change and the further it deviates from the environmental changes seen during model training. Our results in the LIBERO

Table 2: Ablation study of VLA-Rolling inference components with $\pi_{0.5}$ on LIBERO, MetaWorld, and LIBERO-Plus.A, B, C denotes different components of our model. **A**: Action Tail Inheritance; **B**: Confidence-Guided Renoising; **C**: Adaptive-Step Denoising. B_{fixed}: Fixed 50% of renoising. The full VLA-Rolling model corresponds to A+B+C.

Method	LIBERO		MetaWorld		LIBERO-Plus	
	SR (%)↑	Cost (ms)↓	SR (%)↑	Cost (ms)↓	SR (%)↑	Cost (ms)↓
$\pi_{0.5}$	96.38	166.04	58.55	217.28	84.73	161.92
Variants	ΔSR (pp)↑	ΔCost (%)↓	ΔSR (pp)↑	ΔCost (%)↓	ΔSR (pp)↑	ΔCost (%)↓
$+A$	-2.13	+0.02%	**+9.30**	+0.01%	-4.89	+0.02%
$+A + B_{\text{fixed}}$	+0.20	+0.02%	**+9.28**	+0.02%	-0.16	+0.02%
$+A + B$	+0.37	+0.55%	**+9.07**	+0.42%	-0.22	+0.84%
$+A + C$	-1.48	**-22.19%**	**+8.05**	**-15.93%**	-4.29	**-11.91%**
Full Model	**+1.45**	**-23.03%**	**+9.33**	**-16.74%**	**+0.69**	**-12.07%**

Table 3: Observation shift and confidence statistics under different execution horizon K. Correlation values ρ are Pearson correlations computed within each K group. c_{min} is the average minimum of predicted confidence.

Execution Horizon K	Δo_t	Δq_t	c_{min}	$\rho(\Delta o_t, c_{min})$	$\rho(\Delta q_t, c_{min})$
K =10(default)	10.544	0.895	0.359	−0.257	−0.237
K =20	16.071	1.610	0.175	−0.318	−0.207
K =30	19.249	2.151	0.146	−0.272	−0.200
K =40	20.658	2.551	0.140	−0.100	−0.155

table Table 3 show a negative correlation between the drasticness of environmental changes and action confidence, consistent with our assumptions about action confidence in Section 3.2.

As K increases, the correlation between confidence and environmental changes does not decrease significantly until environmental changes far exceed what the default settings (K=10) can comprehend. This indicates that action confidence estimator possesses some robustness, but still has limitations.

Confidence Discrimination Test. We compared the average number of denoising steps for successful and failed tasks on MetaWorld. Figure 5 shows the results, revealing a significant difference in the number of denoising steps: across all task difficulty levels, successful tasks consistently required more denoising steps than failed tasks, up to 2.33 steps. This result indicates that confidence predictions significantly differentiate between successful and failed trajectories, and that denoising budget allocation is consistently related to action generation difficulty.

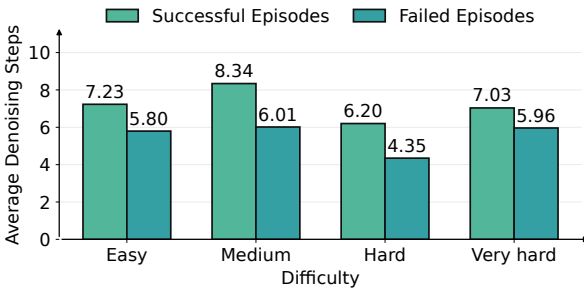
4.4 Real-World Experiments

In the real world, we used $\pi_{0.5}$ on the AgileX PiPER 6-DoF robotic arm to complete our designed target mutation task. In this task, the robotic arm needs to place potatoes into a plate. During the task, we will change the color of the target after the robotic arm approaches the target's placement position. The robotic arm then needs to continue the task to test the model's response to sudden environmental changes after VLA-Rolling is deployed. Detailed settings are shown in the Appendix. E.1.

Table 4: Model performance on real-world experiments.

Model	SR (%) ↑	Cost (ms)↓
$\pi_{0.5}$	70%(14/20)	227.28
+VLA-Rolling	75%(15/20)	176.50

The results in Figure 6 show that the model initially performs the task of *"putting the potato into the blue plate"* normally. However, when encountering a sudden environmental change, the confidence of the action confidence estimator prediction jumps, indicating that the confidence itself is sensitive to the environmental change. Since a decrease in confidence temporarily reverts the model



Difficulty	Successful/ Failure
Easy	350/70
Medium	87/78
Hard	48/42
Very hard	24/51

Figure 5: **Left: Average denoising steps across task difficulty levels on MetaWorld.** Successful trajectories consistently require more denoising steps than failed trajectories. **Right: Distribution of episodes across different difficulty levels**

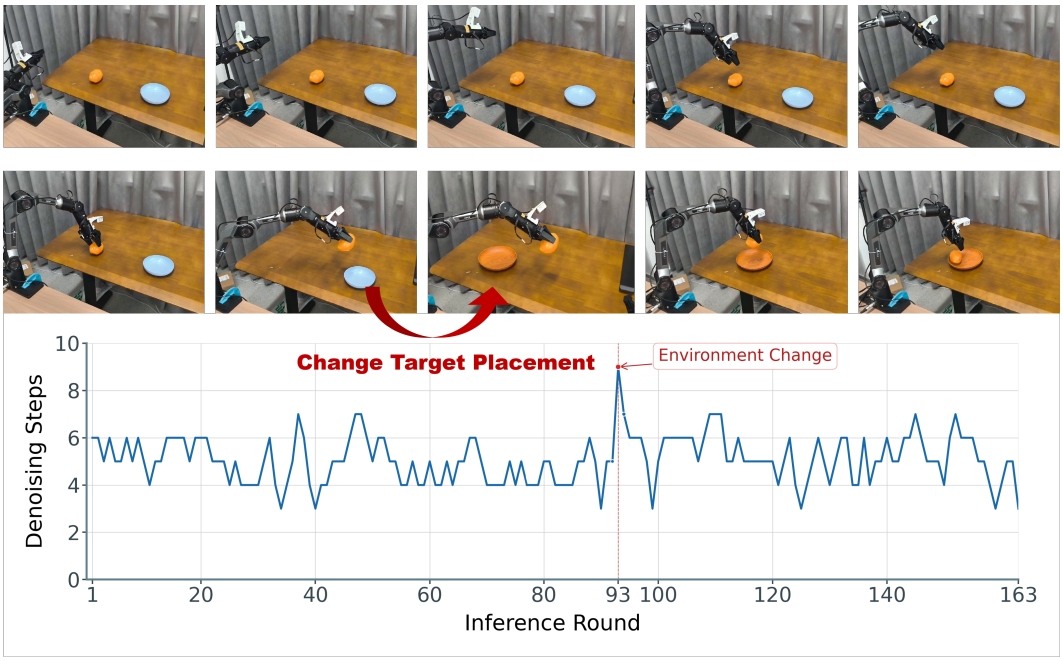


Figure 6: **Real-world experiment of context mutations** revealed a significant decrease in confidence at the mutation site, leading to an increase in the number of denoising steps.

to the original inference paradigm, and because the changed environment is still the same task seen during training, the model can automatically switch to the action of placing the item into the brown plate, demonstrating the practical significance of the renoising process of VLA-Rolling in the task execution process.

5 Conclusion

This paper investigates the reuse value of action tails that are wasted between adjacent inference rounds in the traditional VLA paradigm. We introduce VLA-Rolling, which uses a lightweight plugin to estimate the confidence of action tail, finding the optimal number of denoising steps to guide the reuse process of action tails. Our results show that VLA-Rolling can effectively predict the number of denoising steps required for action tails, efficiently generate high-quality actions, and reuse the motion information in action tails in a rolling manner in each round of inference.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting selected sections of the paper and to improve writing clarity and style. We reviewed and revised all AI-assisted content. Additionally, we used generative AI tools to assist with implementing non-core code components, such as parameter sweeps for ablation studies. All code directly related to the proposed method was reviewed and verified for correctness by multiple authors before use. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We've submitted part of core code in the Supplementary Material to demonstrate the operational logic of our method. The code is constructed based on Lerobot, without requiring intrusive modifications to the Lerobot source code. The complete code will be released in a future open-source release.

REFERENCES

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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.

Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

Jiahong Chen, Jing Wang, Long Chen, Chuwei Cai, and Jinghui Lu. Nanovla: Routing decoupled vision-language understanding for nano-sized generalist robotic policies. *arXiv preprint arXiv:2510.25122*, 2025a.

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*, 2025b.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Yuzhi Huang, Weijue Bu, Ziyi Xiong, Jie Wu, Fanding Huang, Jingyan Jiang, and Zhi Wang. Chainvla: Chaining vision-language-action queries through a unified execution state for long-horizon manipulation. *arXiv preprint arXiv:2608.02326*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Jindou Jia, Tuo An, Yuxuan Hu, Gen Li, Jingliang Li, Bohan Hou, Xiangyu Chen, Jiaqi Bai, Bofan Lyu, and Jianfei Yang. Mars policy: Multimodality only when it matters. *arXiv preprint arXiv:2605.29766*, 2026a.

Jindou Jia, Gen Li, Xiangyu Chen, Tuo An, Yuxuan Hu, Jingliang Li, Xinying Guo, and Jianfei Yang. Action-to-action flow matching. *arXiv preprint arXiv:2602.07322*, 2026b.

Dong Jing, Gang Wang, Jiaqi Liu, Weiliang Tang, Zelong Sun, Yunchao Yao, Zhenyu Wei, Yunhui Liu, Zhiwu Lu, and Mingyu Ding. Mixture of horizons in action chunking. *arXiv preprint arXiv:2511.19433*, 2025.

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.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

Jinhao Li, Yuxuan Cong, Yingqiao Wang, Hao Xia, Shan Huang, Yijia Zhang, Ningyi Xu, and Guohao Dai. Step: Warm-started visuomotor policies with spatiotemporal consistency prediction. *arXiv preprint arXiv:2602.08245*, 2026.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. *arXiv preprint arXiv:2604.04161*, 2026.

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.

Songming Liu, Bangguo Li, Kai Ma, Lingxuan Wu, Hengkai Tan, Xiao Ouyang, Hang Su, and Jun Zhu. Rdt2: Exploring the scaling limit of umi data towards zero-shot cross-embodiment generalization. *arXiv preprint arXiv:2602.03310*, 2026a.

Xuyang Liu, Zichen Wen, Shaobo Wang, Junjie Chen, Zhishan Tao, Yubo Wang, Tailai Chen, Xiangqi Jin, Chang Zou, Yiyu Wang, et al. Shifting ai efficiency from model-centric to data-centric compression. *arXiv preprint arXiv:2505.19147*, 2025.

Yufeng Liu, Hang Yu, Juntu Zhao, Bocheng Li, Di Zhang, Mingzhu Li, Wenxuan Wu, Yingdong Hu, Junyuan Xie, Junliang Guo, et al. Learning native continuation for action chunking flow policies. *arXiv preprint arXiv:2602.12978*, 2026b.

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. Faster: Rethinking real-time flow vlas. *arXiv preprint arXiv:2603.19199*, 2026.

Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. Snapflow: One-step action generation for flow-matching vlas via progressive self-distillation. *arXiv preprint arXiv:2604.05656*, 2026.

Ryuji Oi, Hikari Otsuka, Kosuke Matsushima, Yuki Ichikawa, Masato Motomura, Tatsuya Kaneko, and Daichi Fujiki. Training-free acceleration for vision-language-action models with action caching and refinement. *arXiv preprint arXiv:2607.06370*, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 10832–10851, 2026.

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.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks. *arXiv preprint arXiv:2509.23224*, 2025.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. Vlash: Real-time vlas via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031*, 2025.

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.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. Vla knows its limits: Adaptive execution horizons for robot policies. In *European Conference on Computer Vision*, pp. 607–623. Springer, 2026a.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, et al. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026b.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

Runze Xu, Xiaolong Shan, Shuang Dai, Yu Wang, and Jincheng Yu. Knowing when to stop: Adaptive action chunking via internal cross-attention dynamics in vlas. *arXiv preprint arXiv:2609.00908*, 2026.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40891–40914, 2026.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Dijia Zhan, Xuemiao Xu, Jinyi Li, and Jie Tang. Seam: Smooth execution of action-chunked motion for vision-language-action policies. *arXiv preprint arXiv:2607.04609*, 2026.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713. IEEE, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

Zhide Zhong, Haodong Yan, Junfeng Li, Junjie He, Tianran Zhang, and Haoang Li. Vla-opd: Bridging offline sft and online rl for vision-language-action models via on-policy distillation. *arXiv preprint arXiv:2603.26666*, 2026.

A Related Work

A.1 Continuous Action Generation

In embodied robot control, the VLA paradigm currently dominates (Brohan et al., 2023; O'Neill et al., 2024; Kim et al., 2024; Lu et al., 2026; Pertsch et al., 2025; Black et al., 2024; Zhao et al., 2025; Intelligence et al., 2025; Bjorck et al., 2025; Cai et al., 2026; Wang et al., 2026b; Team et al., 2024), utilizing rich visual language and robot motion information to generate actions. Existing paradigms use flow matching (Black et al., 2024; Bjorck et al., 2025; Intelligence et al., 2025; Wu et al., 2026; Lee et al., 2025; Shukor et al., 2025; Zheng et al., 2026) to transfer the priors of the Vision-Language Model as conditions to guide the generation of continuous future actions. However, only a portion of the generated actions are executed, while the remainder are discarded to prevent error accumulation. Avoiding this computational waste has become a problem that VLA models need to optimize.

A.2 Action Chunk Reuse

Previous work has demonstrated that information from the actions generated by the model can be used to maintain temporal consistency (Black et al., 2026; 2025; Li et al., 2026; Liu et al., 2026b; Sendai et al., 2025; Huang et al., 2026; Oi et al., 2026; Zhan et al., 2026), thereby improving success rate. However, these approaches have obvious drawbacks: they rely too heavily on the quality of the action chunk itself. Low-quality action chunks are still reused as the basis for action generation, leading to incorrect action generation. Therefore, to effectively utilize action chunks, selective utilization is necessary, along with an action-level evaluation mechanism (Jia et al., 2026b;a) within each action chunk, in order to unlock the potential of actions that are discarded from the action chunk.

A.3 Efficient VLA Inference

Efficiency optimizing paradigm for the inference of the Vision-language-Action Model varies. Some of these works attempt to address the issue from the perspective of model size (Yang et al., 2026; Chen et al., 2025a; Wen et al., 2025; Shukor et al., 2025; Wang et al., 2026a), requiring training the model from scratch. However, recently more work is focusing on optimizing the inference paradigm of action decoder, typically resulting in performance loss (Wang et al., 2025; Pei et al., 2026; Liu et al., 2025; Zhong et al., 2026; Luan et al., 2026; Liu et al., 2026a; Tang et al., 2025; Lu et al., 2026). These methods typically sacrifice the accuracy of generated actions for more efficient generation, and cannot avoid making trade-offs between the two. Few works can be universally applied to existing models and produce Patoré improvements.

B Method Details

B.1 Details of Model Architecture

We implement the VLA-Rolling action confidence predictor as an action-centric Transformer with approximately 30M parameters, while keeping the backbone VLA frozen.

The predictor receives the current visual representation and its temporal difference, the current proprioceptive state and its temporal difference, the inherited action tail, and the last executed action.

Visual token sequences are mean-pooled, and the current and difference representations are independently layer-normalized, concatenated, and projected to a 576-dimensional context vector.Finally, a Sigmoid layer is used as the final confidence output layer.

The confidence is then used by VLA-Rolling inference rules to determine the confidence-guided noise and action tail-preserving behavior.

B.2 Details of Action Confidence Estimator

For the simulator environment, we can directly set the execution length of each chunk, thus obtaining an action tail of a fixed length. For action confidence estimator, we strictly align the input and output lengths of its model with the deployed VLA model. For the models we use, except for the X-VLA, which uses $H = 30, K = 10$, and $L = 20$, all other models ($\pi_{0.5}$, SmolVLA, MolmoAct2) use $H = 50, K = 10$, and $L = 40$.

The actual input to action confidence estimator is of the same length as the chunk size H. Each action tail of length $L = H - K$ is padded with Gaussian noise of length K before being fed into the model. During the forward pass, an attention masking mechanism is used to limit the model's focus to the action tail itself, preventing the added Gaussian noise from affecting the model's judgment of action quality. After outputting the result, it is truncated into a confidence block of length L for downstream use.

To avoid having the target VLA model infer across the entire demonstration dataset to obtain the risk quantiles, we made an engineering compromise: during offline training, the $q_{0.1,i}$ and $q_{0.9,i}$ statistics are updated as exponential moving quantiles (EMA) of the valid per-position risks. Specifically, for $p \in \{0.1, 0.9\}$, let $\widehat{q}_{p,i}^{(s)}$ be the p-th quantile estimated from the current training batch at update s; we update

$$q_{p,i}^{(s)} = (1 - \eta) \cdot q_{p,i}^{(s-1)} + \eta \cdot \widehat{q}_{p,i}^{(s)}, \qquad p \in \{0.1, 0.9\}, \tag{11}$$

B.3 Details of Adaptive-Step Denoising

We consider the inherited action tail as a potential intermediate state of the flow-matching denoising trajectory, rather than restarting denoising from the fresh-Gaussian endpoint at flow time $\tau = 0$. Suppose the default VLA denoising process uses $N_{\max}$ steps with step size $\delta = 1/N_{\max}$. For an action tail whose optimal remaining refinement budget is N^*, we regard it as an intermediate state located at the corresponding remaining flow time. Specifically, letting $k = 0, \ldots, N^* - 1$, we initialize

$$\mathbf{A}_t^{(0)} = \mathbf{A}_t^{\text{tail}}, \qquad \tau_k = 1 - \frac{N^* - k}{N_{\max}}, \tag{12}$$

and perform the subsequent refinement using the same Euler update rule as the native VLA:

$$\mathbf{A}_t^{(k+1)} = \mathbf{A}_t^{(k)} + \delta v_\theta \left(\mathbf{A}_t^{(k)}, \tau_k \mid \mathbf{s}_t \right), \qquad \delta = \frac{1}{N_{\max}}. \tag{13}$$

In this formulation, $\mathbf{A}_t^{\text{tail}}$ is no longer treated as a state at $\tau = 0$, but as a partially denoised intermediate state whose starting flow time is determined by its remaining refinement budget N^*. This preserves consistency with the timestep embedding and discretization used in the original VLA denoising process, allowing the action tail to be refined along the remaining portion of the learned flow trajectory.

C Training and Inference Details

C.1 Training Protocal

As for the training EMA rate, we set $\eta = 0.99$. We use AdamW with learning rate 10^{-4} and weight decay 10^{-4}. In our L1 loss for confidence prediction, we adopt $w_+ = 2, w_- = 1$ to realize the asymmetric penalty.

All models we adopted use batch sizes 64 for 4,000 steps. For horizon settings, we use the same default settings as the target VLA model. Training used one NVIDIA RTX PRO 6000 Blackwell Server Edition GPU with approximately 96 GiB of memory.

This online statistic provides a compact estimate of the policy- and horizon-specific risk range without retaining a large offline sample buffer or repeatedly recomputing quantiles over the full dataset. It is therefore an engineering mechanism for efficient deployment of the relative-risk normalization, rather than an additional learned prediction target.

C.2 Inference protocol

Our benchmark tests included LIBERO, MetaWorld, LIBERO-PLUS, and RoboTwin 2.0. LIBERO categorized tasks into four types, including long-range and short-range tasks. LIBERO-PLUS, as its perturbation extension, included 10,030 tasks such as camera viewpoint perturbations, used to verify the robustness of VLA-Rolling. Throughout the testing, we used a VLA model and VLA-Rolling plugin trained only on the original LIBERO demonstration dataset. MetaWorld provided four difficulty levels: Easy, Medium, Hard, and Very Hard. RoboTwin, as the only dual-arm multi-task benchmark among these, had the highest learning difficulty.

During testing, we primarily used two metrics: average success rate as a basic indicator representing model performance, and cost as an efficiency indicator, representing the inference time (in milliseconds) for each chunk. The cost, which only includes the model inference time, consists of three parts: (1) the time for the frozen VLM backbone to generate KV cache for downstream action generation, (2) the time for action expert to denoise, and (3) the forward propagation time of the VLA-Rolling confidence prediction. The reduction in adaptive denoising steps decreases the cost of part (2).

In our main experiment, we tested 100 episodes per task on LIBERO and MetaWorld, 1 episode per task on LIBERO-PLUS, and 100 episodes per task on 50 tasks in the clean subset for Robotwin. All simulator tests were conducted on four RTX 4090 32GB GPUs.

D Additional Experimental Results

D.1 Ablation Sensitivity

We evaluated the sensitivity of VLA-Rolling to the confidence calibration parameter α on the LIBERO dataset. The parameter α controls the slope of the confidence calibration function, thus affecting how prediction confidence translates into action tail preservation and the adaptive denoising budget. We evaluated $\alpha \in \{0.0, 1.0, 1.5, 2.0\}$ while keeping all other parameters constant. Following the same evaluation procedure, we tested all 40 tasks on the LIBERO dataset, 10 epochs per task, and reported the average success rate, average prediction denoising steps, and model inference latency per action block.

The results in Table 5 show that calibration strength affects task performance and computational resource allocation. Without effective calibration (α=0), VLA-Rolling uses an average of 6.861 denoising steps, resulting in the highest inference latency of 206.7 ms/block, with a success rate of only 97.25%. Increasing the α value to 1.0 reduced the average denoising budget to 6.155 steps and inference latency to 187.6 ms/block, but the success rate slightly decreased to 97.00%. This indicates that simply drastically reducing the denoising budget does not necessarily guarantee action quality. At α=1.5, confidence calibration provided a more balanced distribution: the average denoising budget remained at a moderate 6.332 steps, while the success rate increased to 98.50%. Further increasing the calibration strength to α=2.0 did not significantly improve efficiency; instead, it reduced the success rate to 97.25%. However, overall, VLA-Rolling remained relatively stable within the tested calibration range, consistently maintaining a success rate above 97%.

Of all settings, α=1.5 achieved the highest success rate of 98.50% with an average of only 6.332 denoising steps, achieving the optimal balance between action quality and inference efficiency. Therefore, we used $\alpha = 1.5$ as the default calibration parameter in all major experiments.

Table 5: Sensitivity of VLA-Rolling performance to the calibration parameter α on LIBERO. We test 40 tasks with 10 episodes per task.

α	Success Rate (%)	Average Denoising Steps	Model Inference (ms/chunk)
0.0	97.25	6.861	206.7
1.0	97.00	6.155	187.6
1.5 (default)	**98.50**	6.332	195.7
2.0	97.25	6.349	190.9

Table 6: Offline trajectory-level action error under different denoising step convert strategies.

Benchmark	variation	Mean Trajectory Error	Std.	Avg. Denoising Steps
LIBERO	min	53.739	17.311	6.33
	mean	54.420	17.964	4.36
	max	55.989	15.979	2.77
MetaWorld	min	6.046	2.772	7.23
	mean	6.165	2.909	5.63
	max	6.585	2.953	4.12

D.2 Ablation on Denoising Step Convert

We tested different ways of converting confidence to denoising steps and calculated their trajectory-level cumulative errors. The results in Table 6 showed that, overall, taking the minimum confidence as the basis for denoising steps better meets our need to avoid sacrificing accuracy, so we selected it as the default setting.

D.3 Ablation on action confidence estimator parameters

We investigate the impact of VLA-Rolling model capacity by varying the size of the confidence estimator, while keeping the VLA backbone and all other inference settings constant. We evaluate three variants on the LIBERO dataset with approximately 10M, 30M, and 100M parameters, respectively. Following the same evaluation protocol, all 40 tasks were tested, with each task running for 10 epochs. We report the average success rate and its variation across different tasks, as well as the inference cost introduced by VLA-Rolling itself.

As shown in Table 7, increasing the estimator size from 10 million to 30 million improves the average success rate from 97.25% to 98.00%, while the inference cost only increases slightly, from 1.267 ms to 1.579 ms. In contrast, scaling the model to 100M parameters did not yield additional performance improvements; instead, it increased inference cost to 1.731 ms. This indicates that a moderate model size is sufficient to capture the relationship between inherited action tails and the updated execution context, and further increasing the model size provides limited benefits. It also demonstrates that the effectiveness of VLA-Rolling primarily stems from its action tail reuse mechanism, rather than simply increasing model size. Therefore, we used a 30M parameter configuration as the default VLA-Rolling model in all major experiments.

D.4 Ablation on N* Nearby Permutation

To further examine whether the adaptive denoising budget obtained from the VLA-Rolling confidence linear mapping reflects the actual refinement requirements of the inherited action tail, we perturbed the prediction step size by ± 1. As shown in Table 8, both perturbations consistently degrade task performance. On LIBERO, changing the predicted budget to $N-1$ and $N+1$ reduces the success rate by 0.83 and 1.08 percentage points, respectively. A similar trend is observed on MetaWorld, where the corresponding drops are 1.21 and 0.61 percentage points. This local sensitivity suggests that the budget predicted by VLA-Rolling is closely aligned with the refinement demand of individual action tails, rather than behaving as an arbitrary reduction in denoising steps.

Table 7: Performance and inference cost of VLA-Rolling models on LIBERO. Success rate statistics are computed across 40 tasks, 10 episodes per task

VLA-Rolling Model	Success Rate (%)	VLA-Rolling Cost (ms)
10M	97.25 ± 5.47	1.267 ± 0.105
30M	98.00 ± 5.10	1.579 ± 0.335
100M	97.25 ± 7.41	1.731 ± 0.047

Table 8: **N* Locally Optimal Test.**

Benchmark	$N-1\uparrow$	$N\uparrow$	$N+1\uparrow$
LIBERO	97.00 (-0.83)	**97.83**	96.75 (-1.08)
MetaWorld	66.67 (-1.21)	**67.88**	67.27 (-0.61)

D.5 Environmental similarity statistics

These statistics indicate that refreshing observations after performing $K = 10$ steps typically does not correspond to entirely different environmental states. While rolling in the time domain remains necessary to incorporate newly observed biases and prevent error accumulation, the high similarity between adjacent contexts suggests that previously generated action tails are not necessarily completely invalidated after replanning. This provides an empirical basis for VLA-Rolling, enabling the selective reuse of information contained in action tails, rather than unconditionally discarding them and regenerating the entire action chunk from new Gaussian noise.

E Real-World Details

E.1 Real-World Experiment Setting

In our real-world experiments, we chose to use the open-source `pi05-base` from Lerobot as the base model, the AgileX PiPER 6-DoF robotic arm as the end effector, and designed a target placement mutation task to address the confidence dependence on environmental changes.

This task consists of two phases: the robotic arm grasps a potato and attempts to place it into a blue plate. When the potato approaches the upper plate, a perturbation is introduced, removing the blue plate and replacing it with a brown plate. The robotic arm must continue its action and place the potato into the brown plate to complete the task.

We deliberately ambiguous the language instruction for this task as "place the potato into the plate" without specifying the color, allowing both modes to share the same linguistic prior. We then operated the robotic arm to collect 20 trajectories each from the blue and brown plates as target placement points. First, we fine-tuned the `pi05-base` model on the target data, and then used the fine-tuned pi05 model as the frozen VLA backbone to train action confidence estimator.

E.2 Real-World Training Protocol

During real-world deployment, the adaptation of asynchronous execution and VLA-Rolling needs to be considered: the execution horizon of asynchronous execution is determined by both inference latency and control frequency. However, under the control of adaptive denoising steps, our inference latency is not a constant value. Since the time consumed by flow matching accounts for a relatively fixed proportion in the denoising process and is basically linearly related to the number of denoising steps, the inference latency can be estimated from the inference time of its various components.

According to the definition of Table 1, we can formally calculate the inference latency C as:

$$C_{max} = C_{FM} + C_{VLARP} + C_{non-FM} \tag{14}$$

$$C_{min} = \frac{1}{N_{full}} \cdot C_{FM} + C_{VLARP} + C_{non-FM} \tag{15}$$

Table 9: **Observation and proprioceptive-state similarity across a 10-step execution horizon.** We compare the source context at time t with the updated context at time $t + 10$. **Image Cosine** denotes the cosine similarity between the two visual representations; **Image** $(1 - \mathrm{MAE})$ denotes one minus the mean absolute error between the normalized visual representations; **State Cosine** measures the cosine similarity between the corresponding proprioceptive states

Benchmark	Image Cosine	Image $(1 - \mathrm{MAE})$	State Cosine	Standardized State Distance
LIBERO	0.953 ± 0.028	0.951 ± 0.019	0.998 ± 0.003	0.309 ± 0.222
MetaWorld	0.992 ± 0.007	0.986 ± 0.009	0.991 ± 0.012	0.506 ± 0.307

where C_{FM} and C_{VLARP} denotes the forward computation cost of Action Expert and action confidence estimator, C_{non-FM} denotes the cost of the rest of parts.

Subsequently, we can estimate the approximate distribution of the execution horizon:

$$K_{min} = \lfloor C_{min} \cdot f_{control} \rfloor \qquad K_{max} = \lceil C_{max} \cdot f_{control} \rceil \tag{16}$$

where the $f_{control}$ denotes the control frequency.

For general models, the upper and lower bounds of this distribution will not exceed 100ms in physical time, making it difficult to generate excessive environmental changes. Therefore, we choose to construct data pairs for training on these execution horizons. Finally, we train for one epoch and put it into use.

F Limitation and Future Work

This work has limited exploration of cross-environment generalization. Experiments on real robots can be extended to more complex task settings to explore factors beyond environmental abrupt changes that can influence confidence estimation, providing more evidence to explain the mechanism of confidence prediction. Broadening real-world evaluation across additional embodiments and interaction settings is a natural next step. Furthermore, the combination of our inference paradigm with other inference acceleration mechanisms is worth exploring, as this work does not require any modification to the original model and has the potential to accelerate model inference in conjunction with other mechanisms.