POLICY-ALIGNED PREDICTIVE VERIFICATION FOR ADAPTIVE VLA ACTION-CHUNK EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) policies generate action sequences that reduce repeated inference, but deciding how much of each sequence to execute is critical to task success. Existing adaptive methods make this decision using generation stability or prediction agreement, yet these signals do not establish whether executing a candidate prefix preserves the ability to complete the task. We introduce a *policy-aligned predictive verifier* (PAV) that anticipates the consequences of candidate action prefixes and evaluates their compatibility with successful continuation before execution. Rather than using future prediction to generate actions, our method evaluates existing proposals in the VLA's internal task-conditioned representation space. To learn local judgments from terminal task outcomes, we combine trajectory-level survival supervision with a localization prior that preserves useful behavior within failed rollouts. The resulting validity scores guide execution-horizon selection. Evaluations with π_0 and $\pi_{0.5}$ across LIBERO, RoboTwin 2.0, and real-world manipulation show improved success–query trade-offs. These findings point toward a different role for future prediction in robotics, using it to assess how far to trust a policy's proposed actions rather than to generate new ones.

1 INTRODUCTION

Vision-Language-Action (VLA) models have shown strong performance in generating robot actions from visual observations and language instructions (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Miao et al., 2026). Yet querying a large VLA policy at every control step is costly, while robot behavior requires coordinated motion over multiple steps. Action chunking addresses both demands by generating a sequence of future actions in one query, supporting temporal coherence with fewer policy calls. However, executing an entire chunk without new observations limits responsiveness to environmental changes and allows execution errors to accumulate. Many systems therefore use a shorter *execution horizon* to execute only an initial prefix of each chunk before replanning from new observations. Adapting this horizon is thus central to retaining the efficiency of action chunking while remaining responsive to feedback.

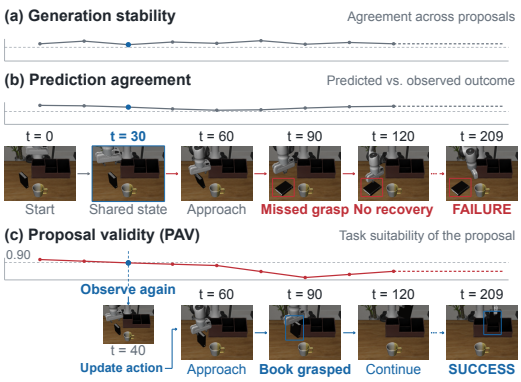


Figure 1: Generation stability (a) and prediction agreement (b) can remain high despite a missed grasp and task failure. Proposal validity (c) guides earlier replanning, leading to successful execution in the illustrated case.

Adaptive execution seeks the longest prefix of each action chunk that still allows the policy to complete the task successfully. Identifying this prefix requires anticipating how the proposed actions will affect the policy's ability to complete the remaining task. Because the effects of a candidate prefix on task success are not directly observable before execution, existing approaches rely on indirect signals to guide execution. Some methods use generation stability as this indirect signal, drawing on internal policy information to select an execution horizon (Wang et al., 2026; Feng et al., 2026; Chen et al.,

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.

2026). Others use prediction agreement, comparing action-conditioned predictions with incoming observations to guide correction or replanning during execution (Pan et al., 2026; Liu et al., 2026). Generation stability and prediction agreement alone, however, do not establish whether a candidate prefix preserves compatibility with successful task completion. As illustrated in Figure 1, an action proposal can be consistently generated and its consequences accurately predicted, yet executing it can still lead to a missed grasp and task failure.

To assess this compatibility directly, we introduce *proposal validity*, which estimates the probability that the policy can successfully complete the task after executing a candidate prefix. This probability depends on the state reached after executing the prefix, because the policy must complete the remaining task from that state. We therefore predict this future state to evaluate whether it supports successful continuation under the policy. World models and World Action Models have demonstrated this capability, building on pretrained visual representations or video models to predict future observations or latent states for planning and action generation (Seo et al., 2022; Du et al., 2023; Zhou et al., 2024; Assran et al., 2025; Ye et al., 2026; Sun et al., 2026). However, because VLA policies already generate effective action proposals, we use this predictive capability to evaluate those proposals rather than to generate new actions. To ground this evaluation in the same information used for action generation, we predict the effects of each prefix in the VLA's task-conditioned representation space, which integrates visual observations and language instructions.

We therefore propose a *policy-aligned predictive verifier* (PAV) that assesses each action extension's compatibility with successful continuation from predicted VLA representations. These extension-level assessments provide local validity estimates used to select execution horizons. To learn these local judgments, however, we must rely on rollouts that provide only terminal task outcomes. These outcomes do not reveal the validity of individual extensions, since even a failed rollout may contain valid action prefixes. We therefore treat extension-level validity as latent and supervise its accumulated product using the terminal outcome. However, the model can fit a failed rollout by slightly lowering validity scores across the rollout, including those of valid extensions. We therefore introduce a localization prior that encourages the model to concentrate these reductions in fewer extensions. At inference time, we combine the local validity scores to evaluate each candidate prefix, accounting for its length. We use these prefix-level scores to decide how many actions to execute before obtaining a new observation and querying the policy again.

Our experiments with π_0 and $\pi_{0.5}$ on LIBERO, RoboTwin 2.0, and real-world tasks assess whether proposal validity selects execution horizons that improve success with fewer policy queries. To examine the basis for these decisions, prediction tests assess action sensitivity, while ablations isolate future prediction, policy alignment, and outcome supervision. Predicting future VLA representations achieves higher success with fewer queries than direct verification or prediction in generic visual features.

2 Related Work

2.1 Adaptive Execution of Action Chunks

Action-sequence policies such as ACT and π_0 generate future action chunks (Zhao et al., 2023; Black et al., 2024). A fixed execution horizon uses the same prefix length across proposals and states, trading fewer queries for later feedback. Adaptive execution instead adjusts when to request a new proposal. One line of work uses internal policy signals to assess generation stability and guide horizon selection. AutoHorizon estimates an execution limit from action self-attention (Wang et al., 2026), while A^3 selects an execution horizon using action consensus and re-decoding stability (Chen et al., 2026). Another line of work assesses prediction agreement from observations during execution. VLA-Corrector truncates and replans when predicted and observed visual dynamics persistently diverge, whereas CheckVLA relates prediction-observation discrepancies to failure risk and latency-aware suffix repair (Pan et al., 2026; Liu et al., 2026). These approaches use generation stability or prediction agreement to guide execution, whereas our verifier predicts the consequences of candidate prefixes and evaluates their compatibility with successful continuation before execution. To our knowledge, we are the first to combine prefix-conditioned future prediction with proposal-validity estimation for adaptive action-chunk execution.

2.2 World Models and Predictive Representations

World models capture action-conditioned dynamics over future observations or latent states (Chua et al., 2018; Hafner et al., 2019b; Hansen et al., 2022; 2024). Dreamer learns behavior from imagined latent trajectories (Hafner et al., 2019a), DINO-WM predicts pretrained visual features for goal-directed planning (Zhou et al., 2024), and World Action Models such as DreamZero jointly predict future video and actions (Ye et al., 2026). Value Prediction Networks evaluate imagined futures through rewards and values (Oh et al., 2017), while Diffuser generates trajectories for planning (Janner et al., 2022). Beyond these uses, prediction can support proposal verification. Given a VLA proposal, future prediction anticipates the consequences of executing candidate prefixes. Assessing these consequences requires representations that retain information relevant to control. DeepMDP and bisimulation-based learning connect representations to rewards and transitions (Gelada et al., 2019; Zhang et al., 2020), while VIP learns visual representations through goal-conditioned value learning (Ma et al., 2022). We therefore predict prefix-induced futures in the action-generating policy's internal representation space, retaining the joint observation and instruction context for outcome-based proposal verification.

2.3 Learning from Trajectory-Level Outcomes

Learning prefix-level validity from terminal outcomes is related to multiple-instance learning, where collection-level labels leave individual instance labels unobserved (Dietterich et al., 1997; Ilse et al., 2018). Here, rollout outcomes supervise temporally ordered prefix extensions without identifying which extension caused a failure. Reward redistribution similarly derives local learning signals from delayed returns or demonstrations (Arjona-Medina et al., 2019; Ren et al., 2021; Patil et al., 2020). A related credit-assignment problem arises in language-model reinforcement learning (Chan et al., 2024), where RED derives token-level rewards from holistic response feedback (Li et al., 2025). These approaches recover local information from global outcomes (Yuan et al., 2024), but we use this supervision to verify proposed execution and select horizons rather than optimize the action-generating policy.

3 Method

3.1 Preliminaries

Action-chunk execution. Adaptive action-chunk execution aims to select the longest prefix likely to preserve compatibility with successful continuation. Given observation o_t and instruction ℓ at decision time t, policy π generates a length-H chunk (Zhao et al., 2023; Black et al., 2024). With $\mathbf{a}_{t,j}$ its j-th action, choosing an execution horizon h from $[H] \triangleq \{1, \dots, H\}$ commits to the prefix

$$\mathbf{A}_t \sim \pi(\cdot \mid o_t, \ell), \qquad \mathbf{A}_t^{1:h} = (\mathbf{a}_{t,1}, \dots, \mathbf{a}_{t,h}), \qquad h \in [H]. \tag{1}$$

After h control steps, the policy receives o_{t+h} and replans. Choosing h therefore balances inference cost and temporal consistency against responsiveness (Wang et al., 2026).

Generation stability as an execution criterion. Because future execution outcomes are unavailable when choosing a horizon, A^3 (Chen et al., 2026) tests whether proposed actions remain stable when other actions in the chunk are fixed as context. Let $\widetilde{\mathbf{a}}_{t,j}$ denote the action regenerated under such a context. This generation-stability signal can be expressed conceptually through the discrepancy

$$e_t(j) = \|\widetilde{\mathbf{a}}_{t,j} - \mathbf{a}_{t,j}\|_2^2 . \tag{2}$$

A small discrepancy indicates that the policy reproduces the proposed action under the imposed context. A^3 therefore accepts the longest prefix that passes its conditional consistency checks. However, consistently generated actions can still prevent later task completion, so generation stability does not establish compatibility with successful continuation.

Prediction agreement as an execution criterion. Prediction agreement provides evidence for intervention by comparing action-conditioned predictions with observations collected during execution (Pan et al., 2026; Liu et al., 2026). To express this signal generically, let $x_{t'}$ denote an observed

feature statistic at control step $t' > t$, $\hat{x}_{t'}$ its earlier prediction, and d a discrepancy measure, giving

$$d_{t'} = d(\hat{x}_{t'}, x_{t'}) \,. \tag{3}$$

A larger discrepancy indicates weaker prediction agreement and provides evidence for correction or replanning during execution. Our decision instead requires evaluating the compatibility of a candidate prefix before the corresponding observations become available.

3.2 Formulating Proposal Validity

Defining the decision target. We therefore evaluate candidate prefixes by their compatibility with successful continuation rather than by generation stability or prediction agreement alone. *Proposal validity* quantifies this compatibility as the probability that successful continuation remains possible throughout the proposed execution. To assess the whole prefix, let $\tau_t^{1:h}$ denote the trajectory induced by the first h actions of a fixed proposal $\mathbf{A}_t$, and let $\mathcal{T}_\ell^{\pi,+}$ contain trajectories from which successful continuation under π remains possible after every extension. We include task progress and remaining time in the state to assess that possibility, and hold the continuation protocol constant to isolate the consequences of the candidate prefixes. Formally,

$$V_t(h) := \Pr\big(\tau_t^{1:h} \in \mathcal{T}_\ell^{\pi,+} \,\big|\, o_t, \ell, \mathbf{A}_t\big) \,. \tag{4}$$

Requiring compatibility throughout execution makes $\mathcal{T}_\ell^{\pi,+}$ prefix-closed, so a compatible trajectory has compatible shorter prefixes. This gives $V_t(h)$ the cumulative structure needed to judge how far execution can proceed while preserving compatibility.

To reduce queries while preserving compatibility, an oracle would select the longest prefix with sufficiently high validity. Let $\mathcal{H} \subseteq [H]$ denote the candidate execution horizons and $\epsilon \in (0, 1)$ a small tolerance. The target horizon is

$$h_t^{\text{target}} = \max(\{\min \mathcal{H}\} \cup \{h \in \mathcal{H} \mid V_t(h) \geq 1 - \epsilon\}) \,. \tag{5}$$

Decomposing proposal validity. To estimate validity before executing progressively longer prefixes, we separate the conditional compatibility of each additional step from the validity accumulated over preceding steps. Since a compatible prefix has compatible predecessors,

$$V_t(h) = V_t(h-1)\, \Pr\big(\tau_t^{1:h} \in \mathcal{T}_\ell^{\pi,+} \,\big|\, \tau_t^{1:h-1} \in \mathcal{T}_\ell^{\pi,+}, o_t, \ell, \mathbf{A}_t\big) \,. \tag{6}$$

We set $V_t(0) = 1$ because no proposed action has yet been executed, and consider the conditional factor where $V_t(h-1) > 0$. Assessing this factor before execution requires anticipating the extension's consequence. We therefore marginalize over its future policy representation z_{t+h} to separate prediction from verification,

$$\begin{aligned} V_t(h) = V_t(h-1) \int & \underbrace{p\big(z \mid \tau_t^{1:h-1} \in \mathcal{T}_\ell^{\pi,+}, o_t, \ell, \mathbf{A}_t\big)}_{\text{prediction}} \\ & \times \underbrace{\Pr\big(\tau_t^{1:h} \in \mathcal{T}_\ell^{\pi,+} \mid z_{t+h} = z, \tau_t^{1:h-1} \in \mathcal{T}_\ell^{\pi,+}, o_t, \ell, \mathbf{A}_t\big)}_{\text{verification}} \, \mathrm{d}z. \end{aligned} \tag{7}$$

Here p denotes the conditional density of z_{t+h}. This decomposition motivates a predictor trained on observed futures and a verifier trained on task outcomes.

3.3 Learning Prediction and Verification

Predicting action consequences in policy space. To approximate the prediction factor in Equation 7, we use a learned point prediction $\hat{z}_{t+h|t}$ for verification, as illustrated in Figure 2. To retain the task context used for action selection, we predict in the policy's representation space, with $z_t = E_\pi(o_t, \ell)$ denoting the features extracted by E_π from the observation and instruction. Encoding observations after executed prefixes supplies targets for the predictor f_θ,

$$z_{t+h} = E_\pi(o_{t+h}, \ell), \qquad \hat{z}_{t+h|t} = f_\theta(z_t, \mathbf{A}_t^{1:h}). \tag{8}$$

Following the use of observed future representations as prediction targets (Schwarzer et al., 2020), we align our predictions with targets from executed transitions $(o_t, \ell, \mathbf{A}_t^{1:h}, o_{t+h})$ collected in

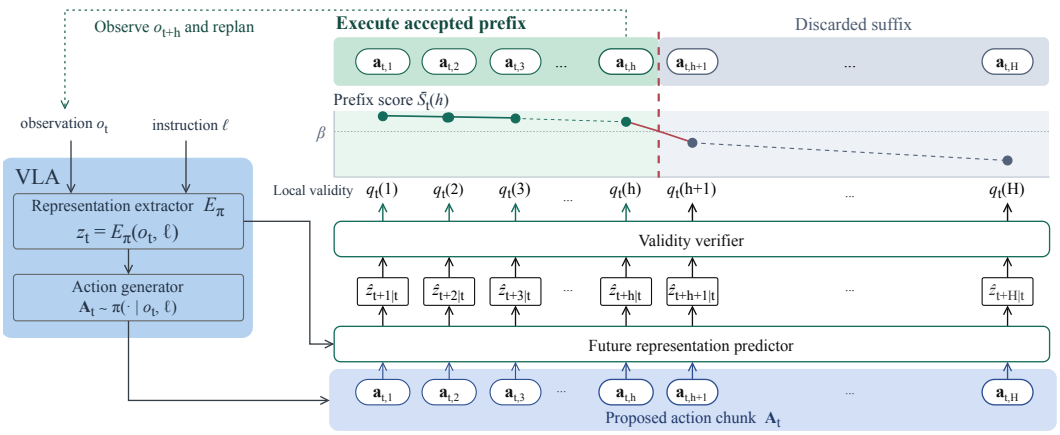


Figure 2: Policy-aligned predictive verification. The predictor uses the VLA representation and proposed actions to predict future policy representations. The verifier estimates local validity, which is accumulated and normalized to select an execution horizon. The robot executes the selected prefix and replans.

$\mathcal{D}_{\text{trans}}$. We use cosine distance $d_{\cos}(u, v) = 1 - \text{sim}(u, v)$, where sim is cosine similarity. Alignment alone may allow the predictor to ignore which actions produced the future. We therefore pair transitions with alternative prefixes $\tilde{\mathbf{A}}_t^{1:h}$ in $\mathcal{D}_{\text{swap}}$, giving $\tilde{z}_{t+h|t} = f_\theta(z_t, \tilde{\mathbf{A}}_t^{1:h})$. Requiring the executed prefix's predicted representation to be closer to the target by margin $m > 0$ encourages action sensitivity without assigning the alternative a failure label. With nonnegative weights $\lambda_{\text{dyn}}, \lambda_{\text{swap}}$ and $[x]_+ = \max(x, 0)$, these requirements give

$$\begin{aligned}\mathcal{L}_{\text{pred}}(\theta) = \lambda_{\text{dyn}}\, \mathbb{E}_{\mathcal{D}_{\text{trans}}}\left[d_{\cos}(\hat{z}_{t+h|t}, z_{t+h})\right] \\ + \lambda_{\text{swap}}\, \mathbb{E}_{\mathcal{D}_{\text{swap}}}\left[\left[m + d_{\cos}(\hat{z}_{t+h|t}, z_{t+h}) - d_{\cos}(\tilde{z}_{t+h|t}, z_{t+h})\right]_+\right].\end{aligned} \tag{9}$$

Learning local validity from trajectory outcomes. Having predicted each prefix's consequence, we assess its compatibility with successful continuation through the verification factor in Equation 7. We use the predicted representation to summarize the context for this conditional judgment and train a verifier q_ϕ from task outcomes. Its output $q_t(h) \in (0, 1)$ at $\hat{z}_{t+h|t}$ estimates local validity. To accumulate these conditional estimates into a prefix score, we follow Equation 6, starting from $\hat{S}_t(0) = 1$,

$$q_t(h) = \frac{\hat{S}_t(h)}{\hat{S}_t(h-1)} = q_\phi(\hat{z}_{t+h|t}),\ \hat{S}_t(h) = \prod_{j=1}^{h} q_t(j), \tag{10}$$

Thus $q_\phi(f_\theta(z_t, \mathbf{A}_t^{1:h}))$ implements the prediction–verification approximation, and its accumulated product estimates $V_t(h)$ without assuming independent extensions.

Since labels are available only for complete rollouts, we extend this accumulation across all policy queries to connect local scores to terminal outcomes. Let $\mathcal{D}^+$ and $\mathcal{D}^-$ collect successful and failed rollouts, with $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$. For rollout τ, let $\mathcal{I}_\tau$ index its executed extensions, $n_\tau = |\mathcal{I}_\tau|$, and q_i denote the corresponding $q_t(h)$, giving $\hat{S}_\tau = \prod_{i \in \mathcal{I}_\tau} q_i$. Success supports validity throughout, so we raise every factor, averaging over n_τ to avoid overweighting long episodes. Failure can coexist with substantial compatible behavior, so we model it as compatibility lost at an unobserved extension and lower only the product. Leaving local attribution latent yields

$$\mathcal{L}_{\text{traj}}(\tau) = \begin{cases} -\dfrac{1}{n_\tau} \log \hat{S}_\tau, & \tau \in \mathcal{D}^+, \\ -\log(1 - \hat{S}_\tau), & \tau \in \mathcal{D}^-. \end{cases} \tag{11}$$

Leaving failure attribution latent avoids labeling every extension as a failure, but the product can still decrease through small validity reductions across the rollout, including at valid extensions. We

Table 1: LIBERO success rates (%) and mean VLA queries per episode. Our method and fixed-horizon baselines use 50 trials per task. † denotes published reference results with potentially different protocols. Bold and underlining indicate the highest and second-highest success rates within each policy setting, including ties. Dashes indicate unreported results.

π_0 $(H = 50)$						
Method	Spatial ↑	Object ↑	Goal ↑	Long ↑	Avg. ↑	Queries ↓
$h = 5$ (Baseline)	**97.4**	97.2	91.0	82.8	92.10	34.6
$h = 10$	96.0	<u>98.4</u>	<u>93.6</u>	84.2	93.05	17.2
$h = 30$	89.4	94.4	90.2	86.6	90.15	6.3
$h = 50$	76.6	75.4	83.4	68.8	76.05	4.4
AutoHorizon (corrected)	95.4	<u>98.4</u>	**94.8**	**87.6**	<u>94.05</u>	10.6
DVAC†	65.0	76.0	76.0	38.0	63.75	6.7
PAV (Ours)	<u>96.4</u>	**98.8**	**94.8**	<u>87.2</u>	**94.30**	12.3
$\pi_{0.5}$ $(H = 10)$						
Method	Spatial ↑	Object ↑	Goal ↑	Long ↑	Avg. ↑	Queries ↓
$h = 5$ (Baseline)	98.6	99.0	<u>98.2</u>	91.4	96.80	32.3
$h = 10$	98.0	<u>99.6</u>	<u>98.2</u>	94.2	97.50	16.1
AutoHorizon	<u>99.8</u>	<u>99.6</u>	97.2	94.6	97.8	27.8
DVAC†	99.0	98.0	98.0	**97.0**	<u>98.00</u>	18.6
A^3	99.2	<u>99.6</u>	97.4	94.0	97.55	16.4
VLA-Corrector†	**100.0**	**99.8**	98.0	93.4	97.80	—
PAV (Ours)	99.0	99.4	**98.4**	<u>95.6</u>	**98.10**	16.8
$\pi_{0.5}$ $(H = 50)$						
Method	Spatial ↑	Object ↑	Goal ↑	Long ↑	Avg. ↑	Queries ↓
$h = 5$	91.8	94.8	84.8	77.2	87.15	38.1
$h = 10$	89.8	92.8	88.0	79.2	87.45	19.2
$h = 30$	68.4	72.4	79.6	69.8	72.55	7.8
$h = 50$	53.2	55.0	72.4	46.6	56.80	5.4
AutoHorizon	<u>95.6</u>	<u>97.2</u>	**92.4**	**92.6**	<u>94.45</u>	12.7
PAV (Ours)	**96.8**	**98.8**	<u>91.8</u>	<u>91.2</u>	**94.65**	14.0

therefore penalize $\sum_i (1 - q_i)$ on failed rollouts. At a fixed product, minimizing this sum favors concentrating the decrease in fewer extensions, allowing high validity elsewhere. With nonnegative weights λ_{traj} and λ_{loc} for outcome supervision and localization, the verification objective becomes

$$\mathcal{L}_{\text{ver}}(\theta, \phi) = \lambda_{\text{traj}}\, \mathbb{E}_{\tau \sim \mathcal{D}}[\mathcal{L}_{\text{traj}}(\tau)] + \lambda_{\text{loc}}\, \mathbb{E}_{\tau \sim \mathcal{D}^-}\left[\frac{1}{n_\tau} \sum_{i \in \mathcal{I}_\tau} (1 - q_i)\right]. \tag{12}$$

Joint optimization. To make predicted representations informative for verification as well as faithful to execution, we jointly minimize $\mathcal{L}_{\text{pred}}(\theta) + \mathcal{L}_{\text{ver}}(\theta, \phi)$. Outcome supervision thus reaches f_θ through the predicted representations used by q_ϕ. Appendix A.2 details the implementation, including label smoothing, minibatch weighting, and auxiliary terminal supervision.

3.4 Execution-Horizon Selection

We apply length normalization to the accumulated validity score to moderate its dependence on prefix length, obtaining $\bar{S}_t(h)$ for horizon selection. Appendix C.5 provides the formulation. We check candidate prefixes in ascending order and stop extending the horizon at the first score below $\beta \in (0, 1)$, giving the execution rule

$$h_t^\star = \max\Big(\{\min \mathcal{H}\} \cup \Big\{h \in \mathcal{H} \;\Big|\; \min_{j \in \mathcal{H},\, j \le h} \bar{S}_t(j) \ge \beta\Big\}\Big). \tag{13}$$

The robot executes the selected prefix and queries the policy from the resulting observation.

Table 2: RoboTwin 2.0 with $\pi_{0.5}$, $H = 32$ (baseline $h = 32$). Success (%) and mean queries cover DVAC's Selected 16 and All 50 tasks. † marks published results, and dashes mark unreported values.

	Selected 16		All 50	
Method	Success ↑	Queries ↓	Success ↑	Queries ↓
$h = 4$	17.38	181.3	27.40	136.5
$h = 8$	46.75	75.0	55.76	54.5
$h = 20$	47.88	28.2	55.16	20.8
$h = 32$	35.38	19.7	43.64	14.6
DVAC†	41.6	—	—	—
PAV (Ours)	**54.13**	59.8	**58.64**	58.5

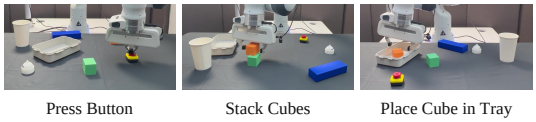
Table 3: LIBERO ablations with $\pi_{0.5}$, $H = 50$. Success is in %, with overall mean queries. PAV matches the setting and results in Table 1.

Suite / metric	Direct	DINOv2	PAV (Ours)
Spatial	83.6	85.4	96.8
Object	91.8	94.2	98.8
Goal	85.0	89.8	91.8
LIBERO-10	77.8	78.6	91.2
Avg. success ↑	84.55	87.00	94.65
Avg. queries ↓	22.3	22.2	14.0

4 Experiments

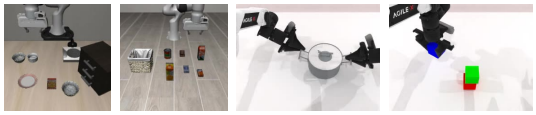
4.1 Experimental Setup

Models and Benchmarks. We evaluate two representative flow-based VLAs, π_0 and $\pi_{0.5}$, to test the execution criterion across action-generating policies (Black et al., 2024; Intelligence et al., 2025). On LIBERO, we use the Spatial, Object, Goal, and Long suites to compare spatial relations, object identity, goal variation, and long-horizon composition in single-arm manipulation (Liu et al., 2023), excluding LIBERO-90. We evaluate π_0 with $H = 50$ and $\pi_{0.5}$ with $H = 10$. To examine a wider range of execution lengths, we also use a $\pi_{0.5}$ checkpoint fine-tuned with $H = 50$, although checkpoint differences prevent isolating the effect of chunk length. We then evaluate $\pi_{0.5}$ with its default $H = 32$ on all 50 RoboTwin 2.0 tasks to test dual-arm manipulation with varied objects and scenes (Chen et al., 2025). Finally, task-fine-tuned $\pi_{0.5}$ policies with $H = 50$ test physical placement, stacking, and contact tasks. Together, these settings assess the criterion across task structures and manipulation setups (Figure 3).



Press Button Stack Cubes Place Cube in Tray

(a) Real-world tasks



LIBERO RoboTwin 2.0

(b) Simulation tasks

Figure 3: **Experimental settings. (a)** Successful real-world task states for Press Button, Stack Cubes, and Place Cube in Tray, shown from left to right. **(b)** Representative simulation environments from LIBERO and RoboTwin 2.0, covering single-arm tabletop and dual-arm manipulation.

Training and implementation details. To expose the verifier to varied execution lengths, we collect rollouts with randomly sampled horizons and balance successful and failed outcomes during training. One verifier is shared across tasks within each policy configuration and benchmark, with evaluation trajectories excluded from training. To isolate method differences, corresponding ablations reuse the same training data, while controlled comparisons share the policy checkpoint and action-generation settings. Appendix A provides the architecture, data collection and splits, optimization settings, and execution parameters.

Metrics. We report success over 50 trials per simulation task and 30 per real-world task alongside mean VLA queries per episode to assess performance relative to inference cost. Each action-chunk generation counts as one query. Selected-horizon distributions characterize execution choices, while verifier computation time measures the added cost of horizon selection.

Table 4: Future-prediction cosine similarity on held-out $\pi_{0.5}$, $H = 50$ LIBERO rollouts, with the context and target held fixed.

Prediction input	Cosine ↑
Executed actions	0.9401
Same-task swapped actions	0.5304
Reversed actions	0.4816
Zero actions	0.4103

Table 5: Action-generation and verifier GPU compute times in ms. Parentheses give verifier time relative to generation.

Setting	Generation	Verifier
π_0, LIBERO $H = 50$	180.34	2.24 (1.24%)
$\pi_{0.5}$, LIBERO $H = 10$	201.19	0.90 (0.45%)
$\pi_{0.5}$, LIBERO $H = 50$	216.95	2.49 (1.15%)
$\pi_{0.5}$, RoboTwin $H = 32$	192.20	2.08 (1.08%)

4.2 Simulation Experiments

4.2.1 LIBERO

Table 1 compares our method with fixed horizons, AutoHorizon, and A^3 using their public implementations (Wang et al., 2026; Chen et al., 2026). Published DVAC and VLA-Corrector results, marked with †, provide additional references for generation stability and prediction agreement (Feng et al., 2026; Pan et al., 2026).

Our method achieves the highest average success across all three policy–chunk settings. The clearest gain over fixed execution occurs with $\pi_{0.5}$ at $H = 50$, where it improves success by 7.50 percentage points over $h = 5$ while reducing queries by 63.3%. At $H = 10$, it raises success from 97.50% to 98.10% with similar query counts, showing benefits even with fewer execution choices. Appendices C.2 and C.5 examine comparisons at similar query counts and the effect of length normalization, respectively.

4.2.2 RoboTwin 2.0

Table 2 compares our method with fixed horizons $h \in \{4, 8, 20, 32\}$ across all 50 tasks and DVAC's 16-task subset for comparison with its published results (Feng et al., 2026). Appendix B provides per-task results.

Our method achieves 54.13% success on the selected 16 tasks, exceeding the best fixed horizon (47.88%) and DVAC (41.6%). This advantage extends to all 50 tasks, where it reaches 58.64% versus 55.76% for the best fixed horizon ($h = 8$), with similar query counts (58.5 versus 54.5).

4.2.3 Analysis of Prediction and Policy Alignment

We first test whether the predictor captures the effects of proposed actions, then compare our method against verification without future prediction (Direct) and prediction in DINOv2 feature space to assess their effects on task success and policy-query cost (Table 3). We conduct these experiments on the four LIBERO suites using $\pi_{0.5}$ with $H = 50$, where ten candidate execution lengths allow us to examine horizon selection over a wider range than the two choices available with $H = 10$.

Does the predictor capture action consequences? For a predicted representation to inform proposal evaluation, it must capture the changes induced by the proposed actions rather than merely preserve the current representation. On held-out rollouts, we therefore compare predicted and observed future latents while holding the current input and observed target fixed. Replacing the executed actions with another same-task sequence tests sensitivity to action content. Copying the current representation tests whether future prediction improves on assuming no latent change. Table 4 compares these alternatives with the prediction conditioned on the executed actions. The prediction conditioned on the executed actions achieves a cosine similarity of 0.9401, compared with 0.5304 for swapped actions.

Does explicit prediction improve execution? To test whether accurate prediction translates into better execution, we compare against Direct, which scores prefixes from current policy representations and actions without predicting future representations. Rollout data, outcome supervision, score accumulation, and horizon selection are held constant. Explicit prediction raises success from

Table 6: Real-world success rates (%), average replanning counts, and average execution horizons. Avg. success is the mean across the three tasks.

Method	Place Cube in Tray ↑	Stack Cubes ↑	Press Button ↑	Avg. success ↑	Avg. replans ↓	Avg. horizon
$h = 5$	16.7	10.0	23.3	16.7	69.58	5.00
$h = 10$	66.7	16.7	40.0	41.1	36.54	10.00
$h = 30$	23.3	43.3	53.3	40.0	11.97	30.00
$h = 50$	26.7	33.3	50.0	36.7	6.91	50.00
PAV (Ours)	**80.0**	**66.7**	**76.7**	**74.4**	**12.87**	**30.22**

84.55% to 94.65% while reducing mean queries from 22.3 to 14.0 (Table 3), supporting its role in execution decisions.

Does the prediction space matter? To test whether policy alignment contributes beyond future prediction, we replace policy representations with DINOv2 features while retaining the prediction–verification pipeline, action-generating policy, and evaluation conditions. DINOv2 achieves 87.00% success with 22.2 queries per episode (Table 3). The higher success and lower query cost in policy space support retaining task-conditioned information for proposal evaluation.

Computational overhead. These gains must also be considered alongside the computation added by predictive verification. Across the four policy and chunk-length settings in Table 5, verifier GPU computation takes 0.90–2.49 ms, or 0.45–1.24% of action-generation time. These measurements quantify GPU computation rather than end-to-end control-loop latency.

4.3 Real-World Experiments

4.3.1 Experimental Setup

Environment. We use a 7-DoF Franka Panda with front, wrist, and right-side RGB observations to provide scene-level and near-gripper views. For each task, we fine-tune $\pi_{0.5}$ from its base checkpoint on 50 teleoperated demonstrations, then collect robot rollouts to train the verifier under the common protocol. Appendix E provides further hardware and training details.

Tasks. The three tasks vary the precision required after transport or approach, allowing us to examine execution under different interaction demands (Figure 3(a)). *Place Cube in Tray* tests transport and placement of an orange cube, while *Stack Cubes* requires placing it on a green cube and makes terminal alignment more restrictive. *Press Button* tests targeted end-effector contact with a red emergency button. To evaluate beyond a single starting arrangement, object positions are randomized within a $50\,\mathrm{cm} \times 50\,\mathrm{cm}$ workspace while remaining visible from the wrist camera.

4.3.2 Real-World Results

Our method achieves the highest success on all three real-world tasks, averaging 74.4% compared with 41.1% for the strongest fixed-horizon baseline, $h = 10$ (Table 6). It also reduces mean replanning counts from 36.54 to 12.87 per episode. Compared with $h = 30$, which has a similar replanning count of 11.97, adaptive execution improves average success from 40.0% to 74.4%, showing the benefit of varying execution lengths rather than using a constant horizon.

5 Conclusion

Effective action-chunk execution depends not only on generating coherent actions, but on deciding how long to commit to them. Our approach uses future prediction to inform this commitment, evaluating proposed execution in the same task-conditioned representation space that guides action generation. Across simulated and real-world manipulation, the results support learning execution decisions from task outcomes rather than relying on consistency alone. More broadly, predictive verification offers a way to improve the use of an existing VLA policy by anticipating the consequences of its proposals before acting on them.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the implementation of methods after the methodology had been independently developed by the authors. We also used generative AI tools to draft parts of the manuscript, improve its readability and presentation, and check grammar. We did not use generative AI tools to develop the research methodology, formulate hypotheses, or interpret experimental results. All AI-assisted text and code were reviewed, revised, and verified by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, we will submit our implementation as supplementary material. The Method section details the proposed approach and training objectives, while the appendix provides the architecture, experimental configurations, and implementation details needed to reproduce our experiments.

REFERENCES

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 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.

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.

Alex J Chan, Hao Sun, Samuel Holt, and Mihaela Van Der Schaar. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.

Feng Chen, Xianghui Wang, Yuxuan Chen, Boying Li, Yefei He, Zeyu Zhang, and Yicheng Wu. Dynamic execution commitment of vision-language-action models. *arXiv preprint arXiv:2605.11567*, 2026.

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.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Xiangdong Feng, Yuxuan Cheng, Chen Shi, Boyao Han, Yuxuan Yan, Yitong Hong, Zhuotao Tian, and Li Jiang. Denoising tells when to replan: Denoising-variance adaptive chunking for flow-based robot policies. *arXiv preprint arXiv:2606.03847*, 2026.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International conference on machine learning*, pp. 2170–2179. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pp. 2127–2136. Pmlr, 2018.

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.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

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.

Jiahui Li, Lin Li, Tai-Wei Chang, Kun Kuang, Long Chen, Jun Zhou, and Cheng Yang. Red: Unleashing token-level rewards from holistic feedback via reward redistribution. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4993–5022, 2025.

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.

Yushan Liu, Peibo Sun, Xintao Chao, Zhenyang Yang, Yifan Xie, Lingfeng Zhang, Shoujie Li, Chenyu Tang, Fang Chen, Xiao-Ping Zhang, et al. Checkvla: Execution-time verification with action-conditioned world model for long-horizon mobile manipulation. *arXiv preprint arXiv:2607.26789*, 2026.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Shangchen Miao, Ningya Feng, Jialong Wu, Ye Lin, Xu He, Dong Li, and Mingsheng Long. Jepa-vla: Video predictive embedding is needed for vla models. *arXiv preprint arXiv:2602.11832*, 2026.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *Advances in neural information processing systems*, 30, 2017.

Yi Pan, Miao Pan, Qi Lu, Jiaming Huang, Man Zhang, Siteng Huang, Xin Li, Jie Zhang, Yongliang Shen, Xuhong Zhang, et al. Vla-corrector: Lightweight detect-and-correct inference for adaptive action horizon. *arXiv preprint arXiv:2607.01804*, 2026.

Vihang P Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M Blies, Johannes Brandstetter, Jose A Arjona-Medina, and Sepp Hochreiter. Align-rudder: Learning from few demonstrations by reward redistribution. *arXiv preprint arXiv:2009.14108*, 2020.

Zhizhou Ren, Ruihan Guo, Yuan Zhou, and Jian Peng. Learning long-term reward redistribution via randomized return decomposition. *arXiv preprint arXiv:2111.13485*, 2021.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pp. 19561–19579. PMLR, 2022.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

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, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

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.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Implementation and Configuration

We detail the $\pi_{0.5}$ LIBERO configuration with a 50-action chunk, no separate text or task embedding, and acceptance threshold $\beta = 0.9$. These settings describe the implementation used for the corresponding analysis setting, rather than assuming an identical architecture or configuration for every policy in the main experiments.

A.1 Architecture of the Predictive Verifier

Policy representation. To retain the context used to generate a proposal, we extract the final prefix hidden states from the policy's PaliGemma language model, before action denoising. Masked mean pooling, masked max pooling, and the last non-padding token each provide a 2,048-dimensional vector. Concatenating these vectors with the policy-preprocessed 32-dimensional state yields a 6,176-dimensional context. The VLA parameters remain unchanged, and the instruction enters through its language tokens rather than through a separate verifier input. In particular, disabling the task embedding removes the additional learned task-ID lookup, not the policy's instruction conditioning.

Predicting this context directly would require a substantially wider dynamics model. We therefore standardize it using training-set statistics and learn a projector from 6,176 to 128 dimensions, followed by ℓ_2 normalization. The predicted and target latents in the method are instantiated in this projected policy-feature space. Current and observed future contexts share the projector, while the future target is detached in the prediction losses. Thus, the policy feature extractor is unchanged, but its downstream projection is learned jointly with prediction and verification.

Action-conditioned prediction. To evaluate all ten candidate horizons in one recurrent rollout, the predictor divides the proposed 50×7 action sequence into ten blocks of five actions. Each block is standardized using training-set action statistics, flattened to 35 dimensions, and embedded into 128 dimensions. Starting from the projected current context and a zero-initialized recurrent hidden state, a GRU cell combines the current latent with the next action embedding. A residual transition head maps its 192-dimensional hidden state to a 128-dimensional update, which is added to the current latent and normalized to obtain the next prediction. This prediction becomes the input latent for the following block. Teacher forcing is disabled during both training and deployment.

Local verification. The local verifier reads only the predicted 128-dimensional future latent and applies a sigmoid to its scalar output to obtain q. It receives neither a separate action embedding nor the GRU hidden state, so action information reaches this score through the predicted consequence. Table 7 specifies the modules. An additional completion head reads the projected observed terminal context during training and is not used to select horizons.

Table 7: Predictive-verifier architecture for $\pi_{0.5}$ on LIBERO with $H = 50$. Widths denote linear-layer dimensions. LN denotes LayerNorm. Dropout is zero throughout.

Module	Layers	Parameters
Context projector	LN, $6176 \to 256$, GELU, $256 \to 128$, ℓ_2 normalization	1,626,560
Action encoder	LN, $35 \to 128$, GELU, $128 \to 128$, GELU	21,190
Recurrent transition	GRUCell, input 256, hidden 192	259,200
Residual update	LN, $192 \to 192$, GELU, $192 \to 128$	62,144
Local verifier	LN, $128 \to 192$, GELU, $192 \to 1$, sigmoid	25,217
Completion head	LN, $128 \to 128$, GELU, $128 \to 1$	16,897
Total	Including the training-only completion head	2,011,208

A.2 Training Data and Optimization

Rollout collection and splitting. To expose the verifier to different execution commitments, we collect 150 rollouts for each of the 40 tasks across the four LIBERO suites, sampling uniformly from $\{5, 10, \ldots, 50\}$ at every policy query. The policy checkpoint is the $H = 50$ LIBERO fine-tuned $\pi_{0.5}$ checkpoint at step 30,000. Observations after each completed five-action block provide future targets, while the terminal task outcome labels the rollout. Unexecuted blocks and incomplete

final blocks are masked out rather than assigned local labels. Future-context extraction uses the same policy prefix encoder as the current context.

To prevent repeated initializations from crossing data partitions, we split initial-state IDs separately within each task and keep all rollouts from an assigned initialization together. With seed 7, this yields 4,200 training, 840 validation, and 960 held-out test rollouts from the 6,000 collected episodes. Normalization statistics are computed only from the training partition. Context statistics include its current, observed future, and terminal contexts, while action statistics are computed per action dimension. Standard deviations below 10^{-5} are replaced by one. Training samples episodes with replacement using inverse class-frequency weights, so success and failure have equal sampling mass.

Prediction and outcome supervision. Future alignment compares each predicted latent with the detached projection of the corresponding observed future. To require sensitivity to the proposed actions, action-swap ranking cyclically permutes action chunks among decision records from the same task within a minibatch. Records without another same-task record are excluded from this comparison. Both predicted representations are compared against the original record's future target, with margin 0.1. The substituted chunk receives no task-failure label.

The trajectory objective treats successful and failed rollouts differently to retain useful local behavior within failures. In this implementation, a successful rollout uses the mean blockwise binary cross-entropy with a smoothed positive target of 0.95, while a failed rollout uses the negative log-probability of at least one incompatible extension. The minibatch objective averages the success and failure class means when both classes are present. Localization penalizes $1 - q$ over all observed blocks from failed rollouts in the minibatch. This implementation uses a block-weighted average, rather than first averaging within each failed rollout and then weighting rollouts equally.

Auxiliary terminal supervision. Because future alignment uses a learned projection, an auxiliary completion objective also trains that representation to retain outcome-relevant information. The completion head takes the projected *observed terminal context* and predicts the rollout's binary success label using class-balanced binary cross-entropy. Its loss is weighted by 0.25 and added to the prediction and verification objectives during joint optimization. It does not predict continuation success from candidate prefixes, and neither its output nor an additional completion factor enters the execution score. The future-contrastive and factorized-completion objectives available in the code are disabled.

Table 8: Verified training settings for the no-task-embedding $\pi_{0.5}$ LIBERO $H = 50$ verifier.

Setting	Value
Collection and training seed	7
Rollouts per task / total	150 / 6,000
Train / validation / test rollouts	4,200 / 840 / 960
Batch size	8 episodes, including their decision records
Optimizer	AdamW, constant learning rate 3×10^{-4}
Training epochs / selected epoch	100 / 7
Checkpoint selection	Lowest validation trajectory loss
Weight decay / gradient norm limit	10^{-4} / 5
Alignment / action-swap weights	1 / 1
Action-swap margin	0.1
Trajectory / localization weights	1 / 0.01
Auxiliary completion weight	0.25
Successful-block training target	0.95
Dropout / teacher-forcing probability	0 / 0
Separate text or task embedding	None

A.3 Candidate Horizons and Execution

To span each chunk with a regular set of execution choices, we use $\{5, 10\}$ for $H = 10$, $\{4, 8, 12, 16, 20, 24, 28, 32\}$ for $H = 32$, and $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$ for $H = 50$.

The five-action prediction stride gives candidate horizons $h_k = 5k$ for $k \in \{1, \ldots, 10\}$. Here k counts predicted extensions, whereas h_k counts robot control steps. To prevent the accumulated score from favoring short prefixes solely through repeated multiplication, deployment applies square-root length normalization to the number of blocks. Writing $q_{t,j}$ for the local score of block j, the score checked at h_k is

$$\widetilde{S}_t(h_k) = \exp\left(\frac{1}{\sqrt{k}} \sum_{j=1}^{k} \log q_{t,j}\right). \tag{14}$$

The implementation clips probabilities to $[10^{-12}, 1]$ before taking logarithms. It scans candidates in ascending order and stops extending the execution horizon at the first score below 0.9. This intermediate-prefix check prevents acceptance beyond an earlier violation, since the normalized score need not be monotone.

The checkpoint's post-hoc bag calibration is not applied to these local scores, and the alternative score obtained by dividing each q by the training target 0.95 is disabled. Table 9 summarizes the resulting deployment configuration.

Table 9: Deployment settings for the verified $H = 50$ configuration.

Setting	Value
Generated action chunk	50×7
Prediction stride	5 control steps
Candidate horizons	$5, 10, 15, 20, 25, 30, 35, 40, 45, 50$
Acceptance threshold	$\beta = 0.9$
Length normalization	Square root of the block count
Candidate check	Stop at the first threshold violation
Bag calibration / target normalization	Disabled / disabled

B Detailed RoboTwin Results

To show how the aggregate results vary across tasks, Table 10 expands the Selected 16 comparison in Table 2 to the complete task list reported in DVAC Table 9 (Feng et al., 2026). Our fixed-horizon baselines and verifier use $\pi_{0.5}$ with $H = 32$, with 50 evaluation episodes per task. The RoboTwin verifier uses $\beta = 0.91$ and includes a task embedding, unlike the LIBERO verifier configuration detailed above. The $h = 32$ baseline comes from a separate evaluation, so episode-level initial conditions are not guaranteed to be paired with the other methods. DVAC is included as an external reference under its published $H = 50$ policy and evaluation protocol.

C Additional Ablations and Analysis

The detailed analysis is organized around $\pi_{0.5}$ with $H = 50$ on LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. With our candidate sets, $H = 10$ allows only $h \in \{5, 10\}$, whereas $H = 50$ offers ten choices from 5 to 50 steps. We use this broader range to examine how the verifier changes execution length and how each component affects that choice. Only the overhead measurements span multiple policy settings. Comparisons use the same policy checkpoint, rollout dataset, task set, and evaluation protocol. These controlled ablations characterize the components in the $H = 50$ LIBERO setting, complementing the broader evaluation of the complete method.

C.1 Future Prediction and Representation Ablations

Table 3 in the main paper reports the complete suite-level comparison of Direct, DINOv2, and PAV. Success and VLA-query counts are shown together to distinguish performance improvements from changes in replanning frequency. Individual loss ablations below further isolate the contribution of the training objectives.

Table 10: Per-task success rates (%) on all 16 RoboTwin tasks in the Selected 16 subset. DVAC† reproduces the $\alpha = 2$ results from its Table 9 (Feng et al., 2026), including the reported average rounded to one decimal place. Other averages are computed over 800 episodes. Bold indicates the highest reported value in each row, including ties.

Task	$h = 4$	$h = 8$	$h = 20$	$h = 32$	DVAC†	PAV (Ours)
Blocks Ranking RGB	0	40	62	38	45	**64**
Blocks Ranking Size	2	34	26	14	37	**50**
Hanging Mug	0	6	**16**	6	1	8
Move Stapler Pad	14	30	22	14	11	**42**
Open Laptop	44	62	84	60	**90**	70
Open Microwave	4	26	46	58	**67**	42
Pick Diverse Bottles	40	**62**	38	36	55	44
Place A2B Left	34	62	64	64	42	**74**
Place A2B Right	22	74	70	56	30	**84**
Place Burger Fries	44	**76**	56	58	58	**76**
Place Dual Shoes	16	48	48	24	27	**56**
Place Empty Cup	26	50	54	58	**68**	56
Place Object Basket	26	60	**68**	28	51	58
Scan Object	4	40	34	12	20	**42**
Stack Blocks Three	0	40	22	10	13	**48**
Stack Bowls Three	2	38	**56**	30	50	52
Selected 16	17.38	46.75	47.88	35.38	41.6	**54.13**

C.2 Acceptance-Threshold Sweep

The threshold changes how much evidence is required before execution and therefore affects both success and query cost. To separate this operating-point choice from representation learning, the sweep holds the trained checkpoint fixed and varies only β. We evaluate $\beta \in \{0.88, 0.89, 0.90, 0.91, 0.92\}$ around the default value of 0.90.

To compare adaptive and constant-horizon execution at similar query counts, we relate Table 11 to the $\pi_{0.5}$, $H = 50$ baselines in Table 1. At $\beta = 0.91$, our method achieves 89.45% average success with 19.7 VLA queries per episode, compared with 87.45% and 19.2 queries for $h = 10$. This 2.00-percentage-point gain supports an advantage at a similar query budget, complementing the default-setting comparison.

Table 11: Threshold sweep with a fixed trained verifier on $\pi_{0.5}$ with $H = 50$ on LIBERO. Success rates are reported per suite and averaged equally across the four suites. Queries and mean horizon are reported overall. The default row uses the PAV results from the main paper.

	Success (%) ↑						
β	Spatial	Object	Goal	LIBERO-10	Avg.	Queries ↓	Mean h
0.88	79.4	87.2	84.6	79.8	82.75	13.8	20.397
0.89	81.8	90.0	85.4	85.4	85.65	15.1	17.233
Default (0.90)	96.8	98.8	91.8	91.2	94.65	14.0	14.98
0.91	89.4	94.6	87.6	86.2	89.45	19.7	11.300
0.92	90.4	94.8	86.8	81.2	88.30	25.1	8.607

To explain how the threshold changes query demand, Figure 4 compares the selected-horizon distributions across the five sweep settings. We normalize the pooled counts within each setting so that differences in the total number of decisions do not obscure the shift in execution length. At the default $\beta = 0.90$, $h = 15$ is the most frequent choice, accounting for 33.0% of selections. The two highest thresholds concentrate execution more strongly on short prefixes, with $h \leq 10$ accounting

for 72.6% of selections at 0.91 and 94.7% at 0.92. The default threshold achieves the highest success among the tested settings (Table 11), while higher thresholds shift selections toward shorter prefixes.

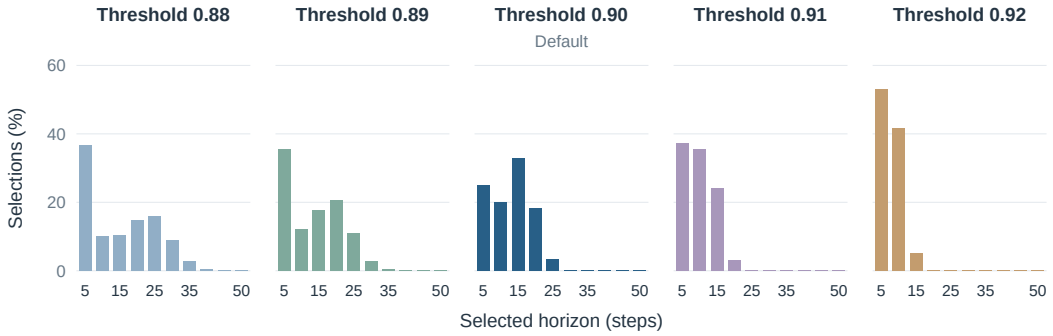


Figure 4: Selected-horizon distributions for $\pi_{0.5}$ with $H = 50$ on LIBERO. Each panel shows one threshold, with counts pooled across the four suites and normalized within that setting. Panels share the same axes, with one bar per candidate horizon. The default setting is highlighted in dark blue.

C.3 Removing Individual Loss Terms

To determine which training constraints contribute to useful execution decisions, each variant removes one term while retaining the architecture, data, and remaining coefficients. Removing future alignment sets $\lambda_{\text{dyn}} = 0$ to isolate the contribution of observed future targets. Removing swap ranking sets $\lambda_{\text{swap}} = 0$ and tests whether alignment encourages sufficient action sensitivity. Removing localization sets $\lambda_{\text{loc}} = 0$ to evaluate how the localization prior contributes to execution performance.

The trajectory objective is retained to provide a common source of outcome supervision. Each trained variant is evaluated with the default threshold, so the comparison measures the contribution of each term under the same execution rule.

Table 12: Loss-term ablations on $\pi_{0.5}$ with $H = 50$ on LIBERO. Suite columns and the average report success (%). Queries are the mean number of VLA generations per episode across the four suites.

Training variant	Spatial	Object	Goal	Long	Avg.	Queries
Full objective	96.8	98.8	91.8	91.2	94.65	14.0
Without future alignment	83.2	91.0	86.8	80.2	85.30	23.99
Without swap ranking	84.6	93.4	88.2	76.6	85.70	24.26
Without localization	81.8	92.2	86.8	73.8	83.65	25.60

The full objective improves success by 8.95–11.00 percentage points over the three ablated variants while reducing mean queries from 23.99–25.60 to 14.0 (Table 12). These results support retaining future alignment, action-swap ranking, and localization in the evaluated configuration.

C.4 Sensitivity to Action Swaps

Table 4 evaluates the trained predictor with executed and same-task swapped actions, alongside reversed and zero-action inputs. We further examine how the effect of replacing the actions varies across prediction horizons. This input-sensitivity diagnostic complements the separately trained loss ablations in Table 12.

A predictor may align with future observations by exploiting scene regularities while responding weakly to the proposed actions. To test this possibility, held-out transitions are evaluated with both the executed prefix and a different prefix of the same length. The current representation and observed

future target are held fixed. Each donor comes from a different held-out episode of the same task with the same number of observed blocks. We retain only recorded future targets with an eligible donor, yielding 36,843 observed-block pairs across 960 test episodes. Each pair receives equal weight.

Using the cosine distance from Equation 9, define the prediction-distance gap

$$\Delta_t(h) = d_{\cos}(\tilde{z}_{t+h|t}, z_{t+h}) - d_{\cos}(\hat{z}_{t+h|t}, z_{t+h}). \tag{15}$$

A positive gap means that the executed prefix predicts its observed consequence more accurately than the alternative. We report the gap by horizon together with the fractions satisfying $\Delta_t(h) > 0$ and $\Delta_t(h) \geq m$. Both inputs are evaluated against the same recorded future, so the comparison measures action sensitivity relative to that target. Across horizons, the executed prefix gives a closer prediction in 89.3–95.2% of pairs, supporting action sensitivity over the tested prediction range.

Table 13: Horizon-wise action-swap sensitivity of the full predictor on held-out transitions. n is the number of observed-block pairs, and the ranking margin is $m = 0.1$.

Horizon	n	Mean $\Delta_t(h)$	$\Delta_t(h) > 0$ (%)	$\Delta_t(h) \geq m$ (%)
5	7,276	0.1628	89.3	53.5
10	6,367	0.3292	92.5	70.7
15	5,537	0.4257	93.6	76.2
20	4,683	0.4898	94.6	79.6
25	3,883	0.5287	94.7	81.0
30	3,160	0.5537	95.2	83.0
35	2,470	0.5619	94.7	82.7
40	1,781	0.5697	94.6	81.9
45	1,140	0.5636	94.0	81.8
50	546	0.5792	95.2	84.6

C.5 Effect of Square-Root Length Normalization

Following Appendix A.3, let k count five-action blocks and $h_k = 5k$ denote the execution length. The accumulated score is $\hat{S}_t(h_k) = \prod_{j=1}^{k} q_{t,j}$, which becomes q^k when local validity is constant. To permit longer prefixes with consistently high local scores while retaining a length penalty, we moderate this decay to $q^{\sqrt{k}}$ using

$$\bar{S}_t(h_k) = \hat{S}_t(h_k)^{1/\sqrt{k}}. \tag{16}$$

This normalized score is used for horizon selection rather than as a calibrated probability. Because normalization can raise later scores after an earlier decrease, the execution rule checks candidate prefixes in ascending order and stops at the first threshold violation. A later increase therefore cannot override an earlier violation.

This ablation tests whether square-root normalization enables longer execution while preserving or improving task success on $\pi_{0.5}$ with $H = 50$ across the four LIBERO suites. Because it acts on the accumulated score at inference rather than as a training regularizer, we hold the trained predictor and verifier fixed and change only the normalization rule. We compare raw accumulation, square-root normalization, and geometric-mean normalization, written as

$$S_t^{(\eta)}(h_k) = \hat{S}_t(h_k)^{1/k^\eta}, \qquad \eta \in \{0, \tfrac{1}{2}, 1\}. \tag{17}$$

Here $\eta = 0$ gives the unnormalized product, $\eta = \frac{1}{2}$ gives our rule, and $\eta = 1$ gives the geometric mean. For constant local validity q, these scores reduce to $q^{k^{1-\eta}}$. Raw accumulation therefore decays exponentially in k, while the geometric mean removes length dependence in this constant-score case. Square-root normalization retains a length penalty but moderates its growth.

All three variants retain the intermediate-prefix acceptance check and the same candidate horizons $h \in \{5, 10, \ldots, 50\}$. Table 14 compares suite-level success, overall query count, and mean horizon at the common default threshold. This comparison evaluates all three rules at the shared operating point.

To explain how normalization changes execution beyond its mean length, Figure 5 shows one selected-horizon histogram per rule, pooling decisions across the four LIBERO suites and normalizing within each variant. Raw accumulation selects only $h = 5$ or $h = 10$, whereas square-root normalization shifts the mode to $h = 15$ and spreads selections across intermediate horizons. Geometric-mean normalization instead concentrates selections at both $h = 5$ and $h = 50$. At the shared threshold, square-root normalization achieves the highest task success and favors intermediate execution lengths, supporting its use as the default normalization in this setting (Table 14).

Table 14: Inference-time normalization ablation with the trained predictor and verifier held fixed. Success rates are reported for each LIBERO suite and averaged equally across the four suites. Queries and mean horizon are reported overall. Entries correspond to the common default threshold.

	Success (%) ↑						
Normalization	Spatial	Object	Goal	LIBERO-10	Avg.	Queries ↓	Mean h
Raw product ($\eta = 0$)	92.0	90.8	91.6	80.0	88.6	34.934	6.529
Square root ($\eta = \frac{1}{2}$)	96.8	98.8	91.8	91.2	94.65	14.0	14.98
Geometric mean ($\eta = 1$)	53.6	53.0	71.8	50.4	57.2	24.735	35.155

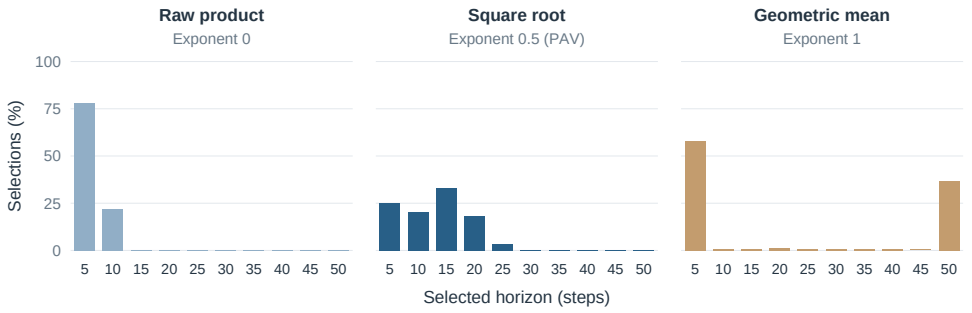


Figure 5: Horizon selection under three normalization rules at $\beta = 0.90$ on $\pi_{0.5}$, LIBERO $H = 50$. Each panel uses identical axes and independently normalized counts pooled across suites. Square-root normalization favors intermediate horizons. Success and query costs appear in Table 14.

D Qualitative Demonstrations

To relate horizon selection to physical interactions, Figure 6 pairs observed rollout frames with execution-horizon traces from LIBERO-Object and LIBERO-Goal. Both examples contain a missed grasp followed by a successful retry and placement. Shorter horizons around the retry provide more frequent opportunities to incorporate feedback, while longer horizons during approach and transfer sustain motion with fewer policy queries. The traces illustrate how execution length changes within a rollout rather than merely differing across tasks.

These examples illustrate shorter commitments during grasp recovery and longer commitments during transfer, complementing the aggregate success and query results. The images show observed rollout frames, and the traces report selected execution horizons.

E Real-World Scene and Demonstrations

E.1 Hardware, Views, and Task Layout

The real-world setup uses a 7-DoF Franka Panda with front, wrist, and right-side RGB observations. Figure 3(a) shows the gripper and tabletop task layouts at successful task states. Object placements vary within the $50\,\mathrm{cm} \times 50\,\mathrm{cm}$ region described in the main text, while remaining visible from the wrist camera at initialization. Camera models, image resolution, control frequency, action interface, inference hardware, and safety settings are *TBD*.

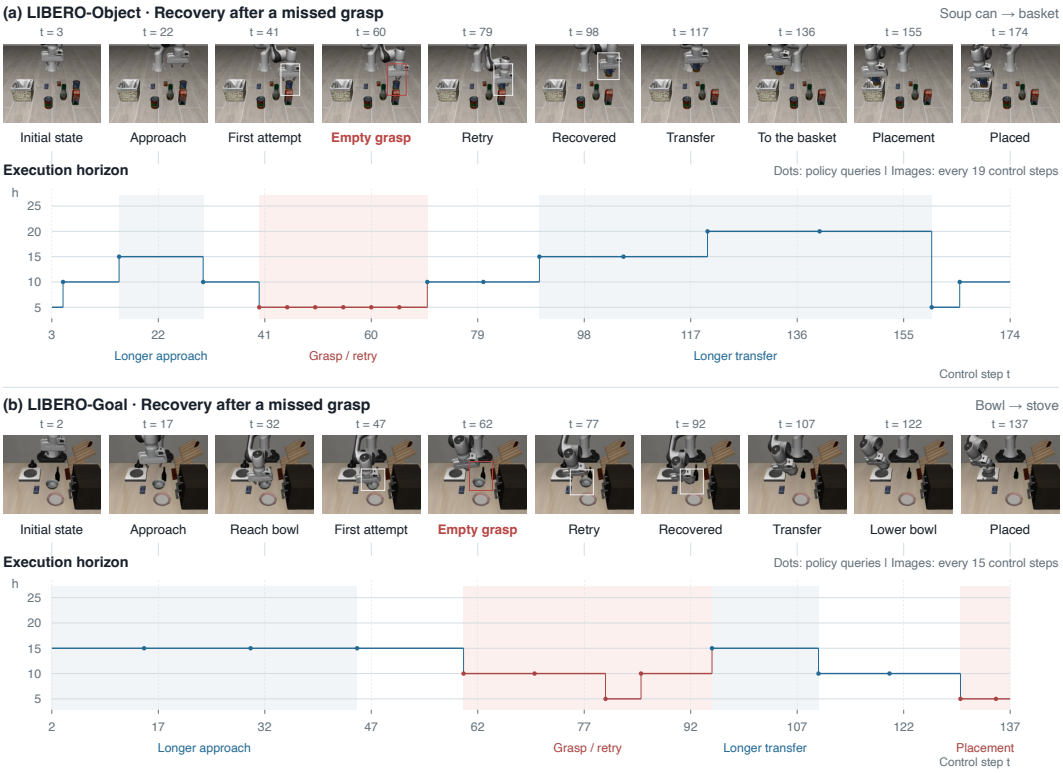


Figure 6: Adaptive execution during grasp recovery in two LIBERO rollouts. (a) In LIBERO-Object, the robot retries an empty grasp before transferring the soup can to the basket. (b) In LIBERO-Goal, it reattempts the bowl grasp and places the bowl on the stove. Observed frames are sampled every 19 and 15 control steps, respectively. The traces show selected execution horizons, with dots marking policy queries. Short horizons accompany grasp retries, while longer horizons occur during approach and transfer. Shaded intervals identify the annotated interaction phases, not validity labels or training supervision.

Task definitions. *Place Cube in Tray* places an orange cube in a tray, *Stack Cubes* places an orange cube on top of a green cube, and *Press Button* activates a red emergency button. The exact success checks, required stability duration, trial timeout, and treatment of aborted trials are *TBD*. These criteria must be fixed across methods so that differences in success reflect behavior rather than scoring rules.

Training and evaluation records. Each task uses 50 teleoperated demonstrations for policy fine-tuning from the $\pi_{0.5}$ base checkpoint. The real-world policies use $H = 50$. The number of fine-tuning epochs, verifier-training rollouts, verifier-training epochs, and the deployed candidate horizon set remain to be specified. Evaluation uses 30 trials per task, separate from the training data. The ordering of methods and the procedure for matching initial object placements are *TBD*.

E.2 Real-World Execution Examples

Figure 7 compares adaptive execution with fixed horizons across three real-world tasks. Adaptive execution combines corrective feedback near interactions with sustained progress afterward. Short execution can interrupt coherent progress, while long execution can continue despite an unsuccessful interaction. Each row shows observations from one rollout; the examples illustrate these behaviors rather than a measured trace of individual policy-query events.

The main real-world table reports average replanning counts and execution horizons alongside success. To compare these measurements with simulation query cost, the final record must specify whether the initial policy call is included in the replanning count. Replanning counts should not be relabeled as total VLA queries without verifying this convention. Likewise, the averaging convention for execution horizons must state whether each executed prefix or each episode receives equal weight.



Figure 7: Real-world qualitative comparisons. **(a) Press Button:** PAV recovers and presses. $h = 5$ oscillates, while $h = 50$ retains an off-target alignment. **(b) Stack Cubes:** PAV realigns and stacks. $h = 5$ grasps but stalls before transfer, while $h = 50$ continues after a missed grasp. **(c) Place Cube in Tray:** PAV returns and retries successfully. $h = 5$ drifts away, while $h = 50$ continues empty before returning later without success. Red annotations highlight key behaviors. All rows start at $t = 0$. Video timestamps are nonuniform and not synchronized across rows.