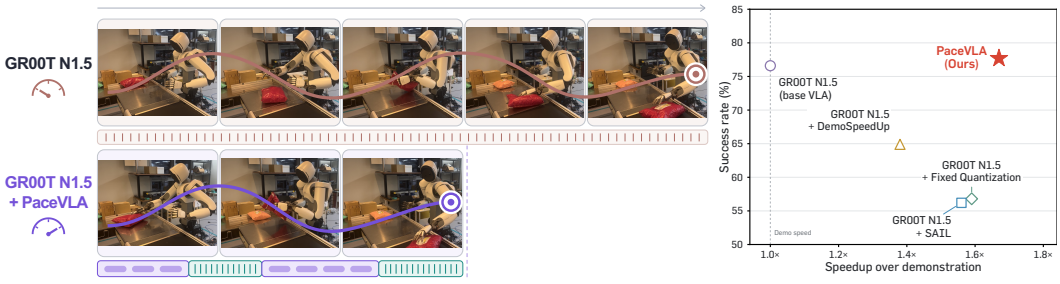
PACEVLA: ADAPTIVE TEMPORAL ACTION QUANTIZATION IN VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fast and reliable manipulation is crucial for practical robot deployment. Although Vision-Language-Action (VLA) models have recently shown remarkable progress as generalist robot policies, their execution often remains temporally inefficient. A key source of this inefficiency is that policies learn fine-grained temporal patterns from teleoperated demonstrations, requiring many low-level action steps. *Temporal action quantization* offers a simple way to accelerate execution by aggregating consecutive low-level actions into temporally coarser actions. However, applying action quantization without considering when fine-grained control is required introduces a speed–accuracy trade-off: accelerated actions may efficiently reduce redundant steps in some phases, while discarding critical motion details in precision-sensitive interactions. To address this, we propose PACEVLA, a unified framework for adaptive temporal execution in VLAs that integrates two coupled components: (1) mode-specific action decoders built on a shared action expert, supporting multiple temporal granularities and prediction horizons, and (2) a quantizability-guided mode selection scheme which decides whether coarse execution is appropriate, and a context-conditioned router that selects an execution mode at each inference step. Experiments across three multi-task simulation benchmarks and six real-world tasks show that PACEVLA improves execution efficiency while maintaining task success rate across multiple embodiments, including long-horizon and dexterous manipulation. The project page is available.



(a) Accelerated execution on the real-robot conveyor task.

(b) Speed–accuracy trade-off.

Figure 1: **Adaptive temporal execution with PACEVLA. (a)** Conveyor package handling task on the humanoid ALLEX: PACEVLA selects a temporal granularity per phase. **(b)** Success rate vs. speedup over the base VLA, averaged over six real-world tasks.

1 INTRODUCTION

Vision-Language-Action models (VLAs) (Zitkovich et al., 2023; Kim et al., 2024; Intelligence et al., 2025; Pertsch et al., 2025; Bjorck et al., 2025) have emerged as a promising framework for generalist robot policies, achieving strong performance through imitation learning from diverse robot demonstrations (Walke et al., 2023; O'Neill et al., 2024; Khazatsky et al., 2024), many of which are collected through human teleoperation (Zhao et al., 2023; Wu et al., 2024). However, teleoperated demonstrations often exhibit conservative behavior: human-in-the-loop control is typically performed through limited-rate interfaces, and human operators adopt cautious motions to avoid failures during data collection (Mandlekar et al., 2018; 2021). Since imitation learning trains policies

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.

to reproduce the demonstrated trajectories, VLAs inherit these patterns and execute tasks through many slow action steps even in phases where faster motion is physically feasible.

Recent works have explored accelerating robot policies beyond demonstration speed, addressing two complementary questions: *where* to accelerate and *how coarsely* to execute actions. DemoSpeedup (Guo et al., 2025) segments per-frame action entropy with a proxy policy and trains a policy on the resulting accelerated trajectories. SAIL (Arachchige et al., 2025) combines high-fidelity tracking with switching between preset slow and fast speeds based on motion complexity and gripper events. A particularly simple mechanism is *temporal action quantization* (Figure AI, 2025; Tang et al., 2025), which merges consecutive actions into coarser ones. It fits naturally with chunk-based VLAs since each predicted action chunk can be executed at a chosen granularity.

Despite their promise, these approaches have two limitations. First, the acceleration decision is either fixed into the training data, using the action entropy of a fine-tuned proxy policy as an implicit indicator (Guo et al., 2025), or applied at execution time to the output of a policy trained at demonstration speed, switching between preset speeds according to kinematic cues (Arachchige et al., 2025). Both signals are statistics of the actions, yet how much precision a phase demands is determined by the robot's interaction with the scene rather than by the motion itself. Second, existing temporal action quantization applies a fixed granularity across the entire trajectory, inducing a speed–accuracy trade-off: coarse actions suffice during free-space motion but discard trajectory details required in precision-critical phases such as grasping and contact-rich interaction. Moreover, granularity is closely coupled with the prediction horizon: phases that tolerate coarse execution can commit to longer predictions, whereas precision-critical phases benefit from shorter horizons with more frequent re-planning. We argue that both the temporal granularity and the prediction horizon of action execution should be treated as a policy decision, adapted to the precision each phase requires.

To this end, we propose PACEVLA, a unified framework for adaptive temporal action execution in VLAs. PACEVLA equips a shared action expert with mode-specific action decoders that support different temporal granularities and prediction horizons. At each inference step, a quantization judge and a router jointly select an execution mode from the visual and language context. Each execution mode is defined by a temporal action quantization level and a prediction horizon, allowing the policy to choose both the granularity of its commands and the temporal extent of its action predictions. To validate the effectiveness of PACEVLA, we conduct extensive experiments across Robocasa-Kitchen (Nasiriany et al., 2024), LIBERO (Liu et al., 2023), and DexJoCo (Wang et al., 2026) simulated robotic benchmarks and six real-world manipulation tasks including long-horizon, dexterous tasks covered on three robot platforms: Franka Research arm with gripper, OpenArm robot arm with two 6-DoF dexterous hands, and humanoid ALLEX robot with two 15-DoF dexterous hands. Compared with the baselines, PACEVLA achieves up to 1.8$\times$ faster than the base VLA, and 1.75$\times$ throughput on the conveyor package-handling task with the humanoid ALLEX, while maintaining task success rates.

Our contributions are:

- We highlight the tendency of VLA policies to inherit fine-grained execution patterns from teleoperated demonstrations as a key source of temporal inefficiency.
- We propose PACEVLA, a lightweight architecture that models execution modes with different temporal quantization levels and action horizons using mode-specific action decoders, a quantization judge and a context-conditioned router.
- We demonstrate consistent gains on simulation benchmarks and real-world tasks, achieving a stronger balance between task success and execution efficiency than the base VLA and fixed temporal quantization baselines.

2 Related Work

Vision-Language-Action models (VLAs). VLAs have recently shown great promise in robotic policy learning by leveraging large-scale robot datasets. One line of VLA models (Kim et al., 2024; Pertsch et al., 2025; Driess et al., 2023) discretizes the continuous action space and predicts actions as tokens, showing that pre-trained VLMs (Beyer et al., 2024; Chen et al., 2026; Bai et al., 2025; Chen et al., 2023) can be effectively adapted to low-level action prediction. In contrast, a growing body of studies (Intelligence et al., 2025; Li et al., 2024) attaches action experts that generate con-

tinuous action chunks via diffusion or flow-matching, enabling more accurate predictions. For example, GR00T N1.5 (Bjorck et al., 2025) and RLDX-1 (Kim et al., 2026) condition a flow-matching head on VLM features to produce action chunks. In this work, we build on a flow-matching action expert for action generation. While most diffusion-based VLAs predict action chunks at a single, fixed temporal granularity set by the data collection frequency, this single granularity is not equally suited to all phases of a task. Motivated by this observation, we explore whether jointly learning actions at multiple temporal granularities and adapting the action granularity to the current context can lead to faster and more accurate task execution.

Faster-than-Demonstration Execution. The execution speed of imitation-learned policies is bounded by the pace of their demonstrations, and several lines of work have aimed to exceed it. Reinforcement-learning-based fine-tuning adjusts the policy using task-level rewards to encourage faster and more decisive behaviors (Yuan et al., 2025), but tends to push the policy toward uniformly faster execution. Another direction modifies the temporal structure of actions, as explored in DemoSpeedup (Guo et al., 2025) and SAIL (Arachchige et al., 2025). While these approaches improve execution speed, they determine the degree of acceleration either at implicit action entropy signals or through kinematic cues for switching between preset speeds, both derived from action statistics, rather than letting the policy itself decide how much to accelerate from the scene it perceives at inference time.

Router-Based Expert Selection. Conditional computation employs a lightweight router that selects which sub-network to apply to each input, and has been widely adopted to scale capacity without proportional cost. In most designs the candidates are structurally identical and differ only in learned weights, whether FFN experts (Fedus et al., 2022) or attention heads (Jin et al., 2024), so a routing decision carries no predefined meaning. Within VLAs, MoLe-VLA (Zhang et al., 2026) routes over backbone layers to reduce the cost of each forward pass, leaving the action output unchanged. Mixture of Horizons (Jing et al., 2025) is the closest to our setting, as it constructs action heads at several chunk lengths, but it fuses their outputs through a gate, so each action is still emitted at a single fixed temporal granularity. None of these, however, treat the temporal granularity of the action as the routing target, where a router selects one decoder per state from a set of decoders that differ in temporal resolution and horizon.

3 METHOD

In this section, we introduce PACEVLA, a unified framework for accelerating robot policy execution through adaptive temporal action quantization. In a nutshell, PACEVLA leverages a shared action expert with multiple action decoders that support different temporal granularities and prediction horizons. At each inference step, a learned quantization judge determines whether to use fine-grained or quantized decoding, and a router selects a mode within the chosen group. We also describe the system-level controller configuration used to track the resulting action commands. The overall framework is illustrated in Figure 2.

Formally, given a language instruction $\boldsymbol{I}$, visual observations $\boldsymbol{V}_t$, and a proprioceptive robot state $\boldsymbol{s}_t$, let $\Phi_t = f_\theta(\boldsymbol{I}, \boldsymbol{V}_t, \boldsymbol{s}_t)$ denote the conditioning context for predicting an action chunk $\boldsymbol{A}_t = (\boldsymbol{a}_{t,1}, \ldots, \boldsymbol{a}_{t,H})$ with action horizon H. We build on a pretrained diffusion-based VLA (Bjorck et al., 2025), whose action expert uses a diffusion transformer (DiT) backbone (Peebles & Xie, 2023) trained with flow matching (Lipman et al., 2022). Given a clean chunk $\boldsymbol{A}_t$, a sampled flow time τ, and Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$, we construct the noisy chunk $\boldsymbol{A}_t^\tau = \tau \boldsymbol{A}_t + (1-\tau)\epsilon$. The velocity network $\boldsymbol{u}_\theta(\Phi_t, \boldsymbol{A}_t^\tau, \tau)$ is trained to predict $\boldsymbol{A}_t - \epsilon$:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\boldsymbol{A}_t, \tau, \epsilon}\left[\left\|\boldsymbol{u}_\theta(\Phi_t, \boldsymbol{A}_t^\tau, \tau) - (\boldsymbol{A}_t - \epsilon)\right\|^2\right]. \tag{1}$$

We extend this formulation to multiple temporal action modes using mode-specific output decoders with a shared action expert.

3.1 TEMPORAL ACTION QUANTIZATION

Temporal action quantization represents a sequence of demonstration actions with fewer action commands by aggregating adjacent actions. We define K action modes. For each mode $k \in \{1, \ldots, K\}$, we convert H_k demonstration steps into $T_k \leq H_k$ commands, with a temporal quantization ratio

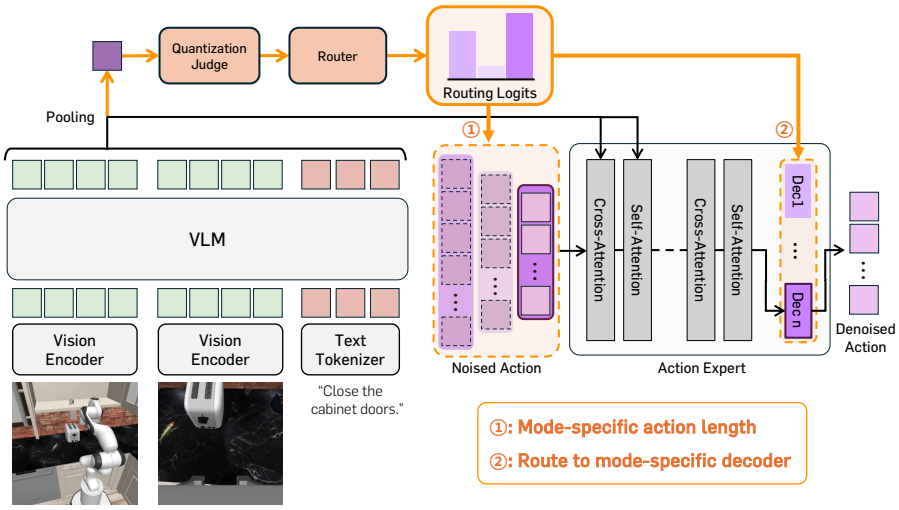


Figure 2: **Overview of PACEVLA.** PACEVLA is a unified framework for adaptive temporal action execution in pretrained VLAs. It equips a shared action expert with mode-specific action decoders that support different temporal granularities and prediction horizons.

$q_k = H_k/T_k$. Here, H_k denotes the source horizon and T_k is the number of predicted commands. Note that *action quantization* refers to aggregating action values along time. We construct fine-grained modes with $q_k = 1$ and quantized modes with $q_k > 1$. Both fine-grained and quantized modes further include shorter and longer source horizons to provide temporal variety.

We partition each source segment into T_k contiguous blocks and aggregate the actions within each block. In detail, action quantization depends on the action representation. For absolute actions, each block retains its final command. For delta-pose actions, we sum translations expressed in a fixed Cartesian frame, compose rotational increments, and retain the final gripper command. The resulting target is denoted by $\boldsymbol{A}_t^{(k)} \in \mathbb{R}^{T_k \times d_a}$, where d_a is the action dimension. Detailed aggregation rules are provided in Appendix A.

3.2 Shared Action Expert with Mode-Specific Action Decoders

We implement these temporal action modes using a shared flow-matching action expert with mode-specific output decoders. The action encoder and DiT backbone are shared across action modes, while each mode k has a lightweight MLP decoder D_k. For mode k, the shared encoder and DiT backbone process a noisy action sequence of length T_k. Decoder D_k maps the resulting features to a flow velocity prediction of shape $T_k \times d_a$, where d_a is the action dimension. This design allows each mode to learn its corresponding temporal granularity without requiring global temporal resampling of the dataset. During training, we compute the loss for all K modes using their respective targets and sequence lengths. At inference, action generation uses the sequence length T_k and decoder D_k associated with the selected action mode.

3.3 Quantizability-Guided Mode Selection

We select an action mode using two signals. The quantization judge first explicitly selects whether to use fine-grained or quantized action, and the router then selects a mode within the chosen group.

Group selection and mode routing. We partition the modes into a fine-grained group, $\mathcal{K}_{\text{fine}} = \{k : q_k = 1\}$, and a quantized group, $\mathcal{K}_{\text{quant}} = \{k : q_k > 1\}$. The predicted quantizability score determines the action group. A lightweight MLP router produces a set of probabilities $\boldsymbol{\alpha}_t = (\alpha_{t,1}, \ldots, \alpha_{t,K})^\top$ over the modes, and these select the mode within the eligible group.

Quantizability supervision. To explicitly determine whether acceleration is appropriate, we supervise the quantization judge with annotations using task execution outcomes and a large vision-

language model (LVLM). We first examine task behavior under several fixed action quantization settings applied to each task. We use these observations as task-level cues to guide the identification of phase-level patterns of quantizability within the demonstrations. LVLM then refines these patterns into criteria using demonstration samples and applies them to each time step of the original demonstrations, producing a per-step quantizability score $y \in [0, 1]$. These scores supervise the judge through eq. (3), while mode-specific prediction errors supervise the router through eq. (2). The details regarding the creation and actual annotation criteria are provided in the appendix C.5.

3.4 Training Objectives

We jointly optimize action prediction and routing, with an auxiliary regression objective for the quantization judge. For mode k, let ℓ_k denote the per-sample prediction error in eq. (1), evaluated using context Φ_t, aggregated target $\boldsymbol{A}_t^{(k)}$, and the velocity network $\boldsymbol{u}_\theta^{(k)}$ formed by the shared action expert and decoder D_k. Each mode uses independently sampled noise and flow times. Prediction errors are masked and averaged over valid action elements, which accounts for the different target action lengths across modes.

Action policy learning. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)^\top \in \mathbb{R}^K$ denote the router probabilities from current context Φ_t, let $\mathbf{w} = (w_1, \ldots, w_K)^\top$ denote its training weights, and $\mathbf{r} = (r_1, \ldots, r_K)^\top \in \mathbb{R}^K$ a target probability vector. The policy objective is

$$\mathcal{L}_{\text{policy}} = \mathbb{E}\left[\sum_{k=1}^{K} w_k \ell_k + \lambda_{\text{sup}} \,\text{KL}(\mathbf{r} \,\|\, \boldsymbol{\alpha})\right] + \lambda_{\text{bal}} \mathcal{L}_{\text{bal}} \tag{2}$$

where the target $r_k \propto \exp(-\ell_k/\beta)$ favors modes with lower prediction errors, with temperature β. The training weights $\mathbf{w}$ mix $\boldsymbol{\alpha}$ with a uniform distribution, and $\mathcal{L}_{\text{bal}}$ penalizes their deviation from uniform, preventing collapse onto a single decoder (Fedus et al., 2022).

Quantizability regression. The judge J_ψ regresses the LVLM-generated quantizability score y at the first frame of each training chunk:

$$\mathcal{L}_{\text{conf}} = \mathbb{E}\Big[\left(J_\psi(\text{sg}(\Phi_t)) - y\right)^2\Big], \tag{3}$$

where sg denotes stop-gradient, so that $\mathcal{L}_{\text{conf}}$ updates only the judge parameters ψ. At inference, group selection uses the judge's predicted score.

The overall training objective is

$$\mathcal{L} = \mathcal{L}_{\text{policy}} + \lambda_{\text{conf}} \mathcal{L}_{\text{conf}}. \tag{4}$$

3.5 Controller Configuration for Accelerated Execution

Temporally quantized actions require controllers to track larger commanded changes. In our experiment, insufficient tracking leads to execution failures. We adjust the tracking and gripper gains to support accelerated execution. Controller-specific settings are reported in Appendix C.3.

4 Experiments

We validate the performance of PaceVLA and the effect of the proposed components through extensive experiments. In particular, we design our experiments to investigate the following questions:

- Does PaceVLA improve the trade-off between task success and execution efficiency on simulation benchmarks with various action representations? (Table 1, 3)
- Can PaceVLA accelerate real-world dexterous and long-horizon manipulation while maintaining task success across different embodiments? (Table 2, 4)
- How does each component of PaceVLA contribute to performance? (Table 5)

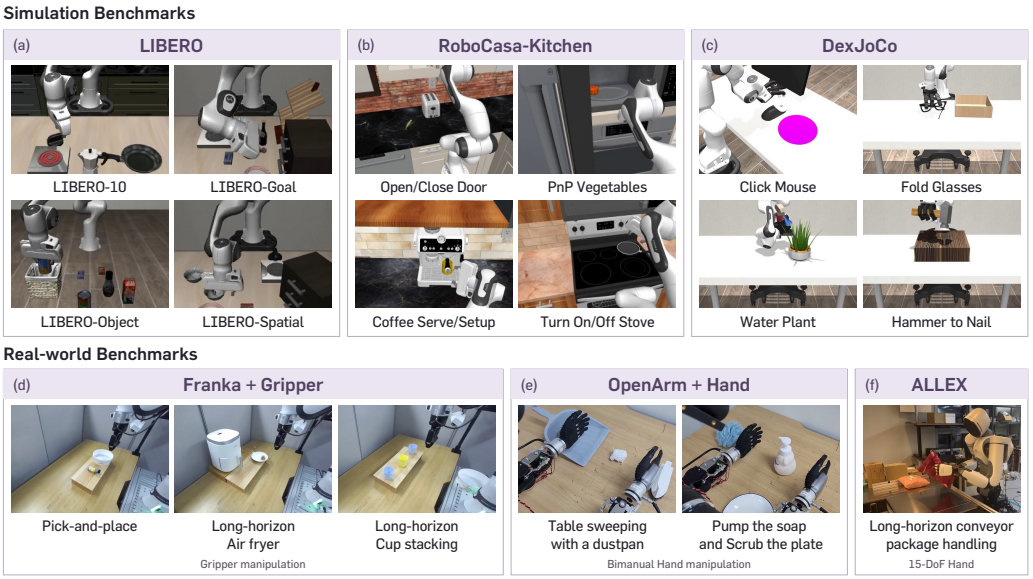


Figure 3: **Experimental setup.** Top: Three simulation benchmarks, LIBERO, RoboCasa-Kitchen, and DexJoCo. Bottom: Real-world manipulation tasks across three embodiments, a Franka Research Arm with a gripper, an OpenArm robot with a dexterous hand, and an ALLEX humanoid. The real-world evaluation includes long-horizon and dexterous manipulation tasks.

4.1 Experimental Setups

Datasets. For simulation, we use RoboCasa-Kitchen (Nasiriany et al., 2024), LIBERO (Liu et al., 2023), and DexJoCo (Wang et al., 2026). We evaluate 24 RoboCasa-Kitchen tasks using 300 demonstrations per task, and report the results separately for pick-and-place tasks and the remaining tasks. For LIBERO, we evaluate the Spatial, Object, Goal, and Long suites. For DexJoCo, we evaluate a subset of six single-arm tasks that require contact-rich dexterous manipulation. In our setup, RoboCasa-Kitchen and LIBERO use delta actions, whereas DexJoCo uses absolute actions, allowing us to assess temporal action quantization under both representations. For real-world evaluation, we evaluate six tasks across three embodiments. On Franka arm with a gripper, we evaluate *Pick-and-Place*, *Air Fryer*, and *Stack Cup*. *Air Fryer* is a long-horizon task that requires opening the basket, placing food inside, and closing it. *Stack Cup* requires two consecutive stacking operations. On a bimanual OpenArm equipped with 6-DoF dexterous hands, we evaluate two bimanual tasks, *Table Sweeping* and *Pump-and-Scrub*. *Table Sweeping* requires sweeping a table using a dustpan and brush, and *Pump-and-Scrub* involves dispensing soap and wiping a plate. On the ALLEX humanoid equipped with 15-DoF dexterous hands, we evaluate long-horizon *Conveyor Package Handling*, in which the robot handles 10 packages in a row per trial, re-orienting each arriving package so that its shipping label faces forward. Detailed task descriptions, datasets, and robot configurations are provided in Appendices C.1 and C.2.

Implementation Details. We implement PaceVLA by fine-tuning pre-trained Vision-Language-Action models (VLA). Specifically, we use GR00T N1.5 (Bjorck et al., 2025) and RLDX-1-IMG (Kim et al., 2026) by following their official implementation. We use four action decoders, corresponding to two fine-grained and two quantized modes, each with a longer and a shorter source horizon, and the quantization judge is supervised using Large-Vision-Language-Model (LVLM)-generated annotations. Training hyperparameters, action mode configurations, annotation procedures, and controller settings are provided in Appendix C.

Baselines. We compare the original model with variants incorporating three acceleration methods and PaceVLA. Action Quantization (Tang et al., 2025) applies fixed temporal action quantization with a ratio of 2 over the entire trajectory on inference. DemoSpeedup (Guo et al., 2025) accelerates demonstrations through entropy-guided temporal downsampling, and SAIL (Arachchige et al., 2025) combines action scheduling, temporally consistent prediction, and speed modulation to sup-

Table 1: **Simulation benchmark results.** We compare PACEVLA with baseline methods on RoboCasa-Kitchen and LIBERO. For RoboCasa-Kitchen, we report results on the pick-and-place tasks (PnP) and the remaining tasks (Others), with models trained using 300 demonstrations per task. For LIBERO, we report results on the short-horizon suites (Short, averaged over Spatial, Object, and Goal) and the long-horizon suite (Long). We report the success rate (SR, %), throughput-with-regret (TPR, $\times 10^{-3}$), and the effective speedup (ES). Base models are listed for reference (ES $= 1$ by definition). Among the acceleration methods for each base model, the best and second-best results are in **bold** and underlined, respectively.

Method	RoboCasa-Kitchen						LIBERO					
	PnP			Others			Short			Long		
	SR ↑	TPR ↑	ES ↑	SR ↑	TPR ↑	ES ↑	SR ↑	TPR ↑	ES ↑	SR ↑	TPR ↑	ES ↑
RLDX-1-IMG [18]	48.5	1.10	1.00	76.3	2.73	1.00	96.7	7.93	1.00	88.6	3.15	1.00
+ Fixed Quantization [30]	43.0	**1.55**	0.87	72.0	**4.00**	0.97	89.4	11.80	1.23	79.2	4.44	1.10
+ DemoSpeedUp [13]	28.7	0.80	0.49	62.7	3.37	0.68	93.5	**14.03**	**1.54**	75.2	**4.95**	1.06
+ SAIL [2]	38.5	1.03	0.72	66.6	2.80	0.75	96.5	10.34	1.27	76.6	3.38	0.92
+ PACEVLA	**45.0**	1.54	**0.93**	**76.9**	3.21	**1.07**	**97.7**	10.36	1.29	**90.2**	4.44	**1.31**
GR00T N1.5 [5]	52.3	1.05	1.00	72.4	2.20	1.00	97.8	8.36	1.00	89.2	3.12	1.00
+ Fixed Quantization [30]	49.0	1.56	0.96	63.1	2.92	0.81	91.2	**12.41**	1.25	82.0	**4.78**	1.21
+ DemoSpeedUp [13]	40.3	1.15	0.71	57.3	2.77	0.67	87.7	12.09	1.14	71.6	3.91	0.90
+ SAIL [2]	32.5	0.77	0.52	58.5	2.31	0.68	95.5	10.85	1.23	79.0	3.89	1.03
+ PACEVLA	**55.0**	**1.64**	**1.16**	**72.9**	**3.30**	**1.14**	**97.1**	10.95	**1.27**	**89.2**	4.27	**1.26**

port faster policy execution. We use GR00T N1.5 (Bjorck et al., 2025) as the backbone for all experiments, and additionally use RLDX-1-IMG (Kim et al., 2026) on the simulation benchmarks to verify that PACEVLA generalizes across VLAs. Baseline adaptations are detailed in Appendix C.6 and evaluation protocols are detailed in Appendix C.1 and C.3.

Metrics. We report the success rate (SR, %) for every task, with partial score on Stack Cup and OpenArm tasks as detailed in Appendix C. In simulation, we report throughput-with-regret (TPR) (Arachchige et al., 2025), which scores each trial by how quickly it succeeds and assigns every failure the same penalty, together with an effective speedup (ES). Following the mean picks per hour (MPPH) metric of Dex-Net (Mahler et al., 2019), we define ES as the ratio of a method's success throughput to that of its base model, where each failed trial is charged the full episode budget $t^{\max}$. A method that moves faster but fails more often can therefore have ES below 1. On the Franka and OpenArm tasks, we report the mean wall-clock execution time (Time, s) over fully successful trials, and on the ALLEX conveyor task, the number of packages handled successfully per minute (Thpt.). Full definitions are given in Appendix B.

4.2 MAIN RESULTS

Simulation Benchmarks. Tables 1 and 3 compare PACEVLA with the acceleration baselines across RoboCasa-Kitchen, LIBERO, and DexJoCo. Across both VLA backbones, PACEVLA largely preserves the success rate of the base policy while improving execution efficiency, achieving an effective speedup above $1\times$ in all but one setting. In contrast, the baseline acceleration methods often attain high raw throughput by sacrificing task success, resulting in an effective speedup below $1\times$ in several settings. This trend is consistent on DexJoCo, where PACEVLA improves both success rate and effective speedup. Overall, PACEVLA provides a more favorable balance between task success and execution efficiency across action representations and VLA backbones.

Real-World Tasks. Tables 2 and 4 compare PACEVLA with the baselines across the three real-world embodiments. On the Franka tasks, PACEVLA reduces execution time by 37–45% while preserving a success rate comparable to or higher than that of the base VLA. Notably, on the Air Fryer task, it reduces execution time from 24.6 s to 14.6 s without degrading success. Fixed Action Quantization achieves comparable or slightly shorter execution on some tasks, but drops substantial success rate by 28.2 and 21.9 percentage points on Air Fryer and Stack Cup, respectively. On the

Table 2: **Real-World evaluation results.** We report average success rate (SR, %, over 32 trials per task; partial scores of 0.5 are given on Stack Cup, Table Sweeping, and Pump and Scrub, see Appendix C) and execution time (Time, average wall-clock execution time on fully successful trials) on five real-world tasks across two embodiments: three Franka+Gripper tasks (*Pick-and-Place, Air Fryer, Stack Cup*) and two OpenArm+Hand tasks (*Table Sweeping, Pump and Scrub*).

	Franka + Gripper						**OpenArm + Bimanual Hand**			
	Pick-and-Place		Air Fryer		Stack Cup		Table Sweeping		Pump and Scrub	
Method	SR ↑	Time (s) ↓	SR ↑	Time (s) ↓	SR ↑	Time (s) ↓	SR ↑	Time (s) ↓	SR ↑	Time (s) ↓
GR00T N1.5 [5]	81.3	12.1	81.3	24.6	84.4	35.3	62.5	21.9	64.1	30.0
+ Fixed Quantization [30]	62.5	7.7	53.1	**14.2**	62.5	**18.7**	48.4	**13.3**	42.2	19.9
+ DemoSpeedUp [13]	68.8	8.1	71.9	18.9	53.1	24.2	56.3	17.9	53.1	20.2
+ SAIL [2]	65.6	7.9	56.3	16.8	45.3	20.0	46.9	15.4	45.3	**15.3**
+ PACEVLA	**84.4**	**7.6**	**81.3**	14.6	**81.3**	19.4	**59.4**	14.6	**65.6**	18.2

Table 3: **DexJoCo benchmark results.** We compare PACEVLA with baselines on DexJoCo, where a dexterous hand performs contact-rich manipulation tasks. We report the success rate (SR, %), throughput-with-regret (TPR, $\times 10^{-3}$), and the effective speedup (ES) over 6 single-arm tasks. Among the acceleration methods, the best and second-best results are in **bold** and underlined, respectively.

Method	**DexJoCo (Single-arm)**		
	SR ↑	TPR ↑	ES ↑
GR00T N1.5 [5]	76.3	1.83	1.00
+ Fixed Quantization [30]	63.0	1.65	0.70
+ DemoSpeedUp [13]	65.7	1.75	0.76
+ SAIL [2]	60.3	1.58	0.65
+ PACEVLA	**77.3**	**2.22**	**1.11**

Table 4: **Real-world evaluation results.** We compare PACEVLA with baselines on Conveyor-based package handling task with a humanoid ALLEX (15-DoF hand). We report the success rate (SR, %) and throughput (Thpt., packages/min). Among the acceleration methods, the best and second-best results are in **bold** and underlined, respectively.

Method	**Conveyor Package Handling**	
	SR ↑	Thpt. ↑
GR00T N1.5 [5]	86.0	3.6
+ Fixed Quantization [30]	72.0	4.2
+ DemoSpeedUp [13]	86.0	4.7
+ SAIL [2]	78.0	4.3
+ PACEVLA	**94.0**	**6.3**

bimanual OpenArm tasks, PACEVLA reduces execution time by 33–39% while keeping the base model's success rate. Finally, on the ALLEX conveyor task, PACEVLA increases throughput from 3.6 to 6.3 packages per minute while improving success rate from 86% to 94%. These results show that PACEVLA transfers across gripper-based, bimanual, and humanoid manipulation without the substantial success rate degradation observed with the baseline acceleration methods.

4.3 MORE ANALYSIS

Ablation Study. First, we remove each of the two key components one at a time. (i) Without the mode-specific decoders, we use the base VLA's single decoder and directly quantize its predicted chunk with the same aggregation rule when the judge selects coarse execution. (ii) Without the quantization judge, the router selects among all modes. As shown in Table 5a, removing the mode-specific decoders reduces the success rate from 66.9% to 63.3% and the effective speedup from 1.14 to 1.00, indicating that dedicated decoders are important for jointly preserving task success and execution efficiency under temporal quantization. This is consistent with Appendix D.1, where dedicated decoders predict quantized actions more accurately than directly quantizing the base model's predictions. Removing the quantization judge lowers TPR from 2.75 to 2.01 and ES from 1.14 to 1.01, showing that explicit quantizability supervision is necessary to decide when coarse execution is appropriate. Next, we examine the loss design in Table 5b. Removing the KL objective causes the largest degradation, reducing SR, TPR, and ES to 63.7%, 2.04, and 0.98, respectively. Removing the balancing objective also degrades all three metrics, showing that both objectives contribute to learning effective mode selection. Lastly, Table 5c compares router input representations. Average pooling achieves the highest SR, TPR, and ES, outperforming both attention pooling and the last-token representation, and we therefore use it as the default router input.

Table 5: **Ablation study.** Success rate (SR, %), throughput-with-regret (TPR, $\times 10^{-3}$), and effective speedup (ES) on RoboCasa-Kitchen, averaged over all 24 tasks. ES is computed relative to the base model GR00T N1.5. Our default setting is highlighted in violet.

Method	SR ↑	TPR ↑	ES ↑
w/o Mode-specific Decoders	63.3	2.44	1.00
w/o Quantization Judge	65.1	2.01	1.01
PACEVLA	**66.9**	**2.75**	**1.14**

(a) Component ablation.

Method	SR ↑	TPR ↑	ES ↑
w/o KL	63.7	2.04	0.98
w/o Balance	64.8	2.38	1.04
Balance + KL	**66.9**	**2.75**	**1.14**

(b) Loss design.

Method	SR ↑	TPR ↑	ES ↑
Attn. Pool	64.6	2.38	1.04
Last Token	64.2	2.43	1.03
Avg. Pool	**66.9**	**2.75**	**1.14**

(c) Router input.

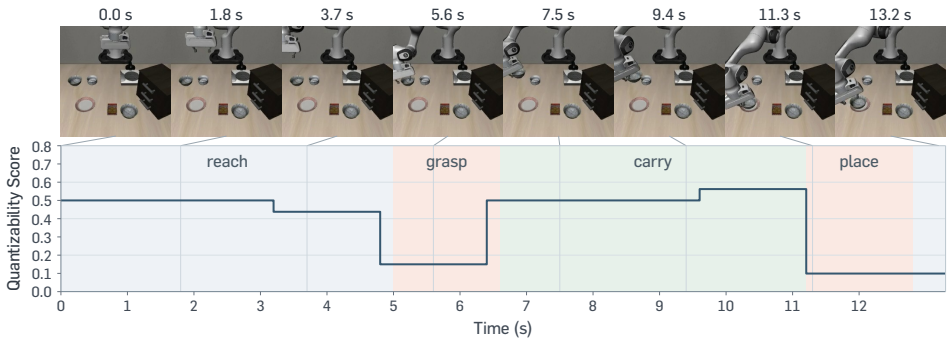


Figure 4: **Quantizability supervision analysis.** We visualize the quantizability scores annotated on a RoboCasa-Kitchen demonstration. Shaded bands indicate manipulation phases. Scores are low around contact transitions (grasp and place) and higher during reach and carry, so coarse execution is suppressed exactly where precise control matters.

Qualitative Analysis. Figure 4 visualizes the quantizability scores assigned along a representative RoboCasa-Kitchen demonstration. The quantizability scores remain higher during reaching and carrying, where coarse execution is generally tolerable, and decrease around grasping and placement, where finer control is required. The score also drops during the final approach to the object, indicating that the annotation captures changes in temporal precision requirements across phases.

Limitation. Table 6 compares the per-step inference latency, averaged over 100 runs on an RTX 5090 GPU. PACEVLA incurs a slight overhead over the base model ($1.05\times$), as the judge and router must run before action generation. However, this $\sim$2 ms cost is negligible compared to the execution time saved (*e.g.*, 24.6 s $\rightarrow$ 14.6 s on Air Fryer), so the efficiency gain far outweighs the added inference cost.

Table 6: **Inference cost.** Per-step latency on RoboCasa-Kitchen.

Method	Latency (ms) ↓	Rel. Cost
GR00T N1.5	40.07	$1.00\times$
PACEVLA	42.15	$1.05\times$

5 CONCLUSION

In this work, we have presented PACEVLA, a unified framework for adaptive temporal action execution in Vision-Language-Action models (VLAs). Motivated by the observation that VLAs inherit the fine-grained execution patterns of teleoperated demonstrations, and that fixed temporal action quantization trades task success for speed, we treat the temporal granularity and prediction horizon of actions as a policy decision rather than a fixed design choice. PACEVLA equips a shared action expert with mode-specific decoders operating at different temporal granularities and prediction horizons, and selects among them at each inference step through a quantizability-guided judge and a context-conditioned router. Our experiments on three simulation benchmarks and six real-world tasks across gripper-based, dexterous-hand, and humanoid platforms show that PACEVLA substantially shortens execution while maintaining the task success of the base VLA, unlike fixed or rule-based acceleration methods, with only marginal inference overhead. We hope that our work facilitates future research toward generalist robot policies that adapt their execution pace to the task at hand, enabling faster and more reliable deployment in practice.

AI USE STATEMENT

In this work, we used generative AI tools as part of the quantizability supervision pipeline. Claude Opus 5 (Anthropic, 2026) was used to help develop and refine the quantizability criteria and annotation prompts based on task-level evaluation outcomes and demonstration examples. Gemini 3.8 Flash (Google DeepMind, 2026) was subsequently used through its API to annotate demonstration chunks according to these criteria. The resulting ratings were aggregated into quantizability scores used to supervise the quantization judge, as described in Appendix C.5. We also used generative AI tools for grammar checking and minor readability edits of author-written text. The authors reviewed the AI-assisted research process and resulting manuscript content and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide hyperparameter details in Section 4 and Appendix C.3. In addition, we provide prompts in Appendix E.

REFERENCES

Anthropic. Claude opus 5 system card. Model card, July 2026. URL https://www.anthropic.com/claude-opus-5-system-card.

Nadun Ranawaka Arachchige, Zhenyang Chen, Wonsuhk Jung, Woo Chul Shin, Rohan Bansal, Pierre Barroso, Yu Hang He, Yingyang Celine Lin, Benjamin Joffe, Shreyas Kousik, et al. Sail: Faster-than-demonstration execution of imitation learning policies. *arXiv preprint arXiv:2506.11948*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Johan Bjorck, Valts Blukis, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1.5: An improved open foundation model for generalist humanoid robots. https://research.nvidia.com/labs/gear/gr00t-n1_5/, June 2025. Accessed: 2025-09-09.

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.

Guo Chen, Zhiqi Li, Shihao Wang, Jindong Jiang, Yicheng Liu, Lidong Lu, De-An Huang, Wonmin Byeon, Matthieu Le, Max Ehrlich, et al. Eagle 2.5: Boosting long-context post-training for frontier vision-language models. *Advances in Neural Information Processing Systems*, 38:91077–91100, 2026.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Figure AI. Helix accelerating real-world logistics. https://www.figure.ai/news/helix-logistics, 2025.

Google DeepMind. Gemini 3.8 flash model card. Model card, September 2026. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/.

Lingxiao Guo, Zhengrong Xue, Zijing Xu, and Huazhe Xu. Demospeedup: Accelerating visuomotor policies via entropy-guided demonstration acceleration. *arXiv preprint arXiv:2506.05064*, 2025.

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.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. Moh: Multi-head attention as mixture-of-head attention. *arXiv preprint arXiv:2410.11842*, 2024.

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.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.

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. In *Conference on Robot Learning*, 2024.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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.

Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science robotics*, 4(26): eaau4984, 2019.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pp. 879–893. PMLR, 2018.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

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 *IEEE International Conference on Robotics and Automation*, pp. 6892–6903. IEEE, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

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.

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.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, et al. Dexjoco: A benchmark and toolkit for task-oriented dexterous manipulation on mujoco. *arXiv preprint arXiv:2605.16257*, 2026.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12156–12163. IEEE, 2024.

David D Yuan, Tony Z Zhao, Kaylee Burns, and Chelsea Finn. Speedtuning: Speeding up policy execution with lightweight reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1184–1192. IEEE, 2025.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18764–18772, 2026.

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.

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

A Temporal Action Quantization Details

The aggregation rule depends on the action representation. For a source segment of H actions quantized into T commands, let $0 = s_0 < s_1 < \cdots < s_T = H$ denote the boundaries of the T contiguous temporal blocks. The i-th block therefore contains $\{\mathbf{a}_{s_{i-1}+1}, \ldots, \mathbf{a}_{s_i}\}$. When H is not divisible by T, we partition the source segment into T contiguous blocks whose sizes differ by at most one. For absolute actions, each block is represented by its final command:

$$\bar{\mathbf{a}}_i = \mathbf{a}_{s_i}. \tag{5}$$

This preserves the endpoint while removing intermediate commands.

For delta-pose actions, directly summing all action dimensions is generally invalid because rotational increments do not compose additively. We write each action as $\mathbf{a}_j = (\Delta\mathbf{p}_j, \mathbf{u}_j, g_j)$, where $\Delta\mathbf{p}_j$ is the translational increment, $\mathbf{u}_j$ is the rotational command, and g_j is the gripper command. Translations expressed in a fixed Cartesian frame can be summed directly, whereas rotations must be composed on $\mathrm{SO}(3)$.

Let ϕ denote the mapping from the policy's rotational command to the rotation vector used by the controller, and define $\mathbf{R}_j = \mathrm{Exp}(\phi(\mathbf{u}_j))$. Here, $\mathrm{Exp} : \mathbb{R}^3 \to \mathrm{SO}(3)$ and $\mathrm{Log} : \mathrm{SO}(3) \to \mathbb{R}^3$ denote the mappings between rotation vectors and rotation matrices. Under the spatial-increment convention, the quantized action for block i is

$$\bar{\mathbf{a}}_i = \left(\sum_{j=s_{i-1}+1}^{s_i} \Delta\mathbf{p}_j,\ \phi^{-1}\left[\mathrm{Log}\left(\mathbf{R}_{s_i} \cdots \mathbf{R}_{s_{i-1}+1}\right)\right],\ g_{s_i} \right). \tag{6}$$

Because rotations are noncommutative, directly adding rotational increments does not in general reproduce their ordered composition. The SO(3) composition in eq. (6) preserves the net commanded rotation of the original block under the controller convention, although quantization may alter the intermediate trajectory.

When actions are normalized for policy learning, we first transform them back to controller coordinates, perform the aggregation above, and then normalize the resulting actions again. The resulting quantized target chunk is denoted by $\mathbf{A}_t^{(k)} \in \mathbb{R}^{T_k \times d_a}$.

B Evaluation Metrics

In this section, we describe the two efficiency metrics used in the simulation benchmarks in detail.

Throughput-with-regret (TPR). Following SAIL (Arachchige et al., 2025), TPR is defined as

$$\mathrm{TPR} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{S_i}{t_i} - \frac{1 - S_i}{t^{\max}} \right), \tag{7}$$

where N is the number of trials, $S_i \in \{0, 1\}$ indicates whether trial i succeeds, t_i is its number of executed action steps, and $t^{\max}$ is the maximum number of steps allowed per trial, after which the trial is terminated and counted as a failure. TPR rewards faster successes while penalizing all failures equally.

Effective speedup (ES). Since TPR penalizes a failure by only $1/t^{\max}$, it underweights the time lost to failed trials. We therefore additionally report

$$\mathrm{ES} = \frac{T_{\mathrm{base}}}{T}, \qquad T = L + \frac{1 - \mathrm{SR}}{\mathrm{SR}} \cdot t^{\max}, \tag{8}$$

where $\mathrm{SR} \in (0, 1]$ denotes the success rate expressed as a fraction, L is the mean number of action steps over successful trials, T is the expected number of action steps per success including the $t^{\max}$ steps consumed by each failed trial, and T_{base} is T of the corresponding base model. Equivalently, $1/T$ is the number of successes per action step, i.e., the success throughput, following the mean-picks-per-hour (MPPH) metric of Dex-Net (Mahler et al., 2019). $\mathrm{ES} > 1$ indicates that a method completes tasks faster than its base model once failures are taken into account, and $\mathrm{ES} = 1$ for the base model by definition.

Evaluation horizon. We use $t^{\max} = 1500$ for RoboCasa-Kitchen and DexJoCo, and $t^{\max} = 220/280/300/520$ for LIBERO-Spatial/Object/Goal/Long, following the default settings.

C Experimental Details

We collect 100 demonstrations for Pick and Place, 50 each for Stack Cup, Table Sweeping, Air Fryer, and Pump and Scrub, and 320 for Conveyor Package Handling across three embodiments: a single-arm Franka Research 3 with a parallel-jaw gripper (Pick and Place, Stack Cup, Air Fryer), a bimanual OpenArm with two 6-DoF hands (Table Sweeping, Pump and Scrub), and the ALLEX humanoid (Conveyor Package Handling). The Franka setup uses a third-person and a wrist-mounted camera, and the bimanual setup uses a stereo egocentric camera pair, all at 720×1280 without depth; the ALLEX setup uses a head-mounted stereo camera as its sole vision sensor. We use absolute joint positions as the action mode: 8-dim for the Franka tasks, 26-dim for the bimanual tasks, and 48-dim for the ALLEX task, alongside joint torque estimates from all 48 DoF as physical signals. Unless otherwise noted, each model is evaluated over 32 trials.

C.1 Task Specifications

Pick and Place The task requires picking up a single target object from the table and placing it into the plate. To prevent memorization of one grasp pose, we vary the target object across three categories that differ in size, shape, and required grasp width and vary the object position across the workspace to evaluate spatial generalization. We evaluate each model over 32 trials, consisting of 10,11, and 11 trials per object. We use the instruction "Pick up the [doll / green cube / blue cup] and put it into the plate." for this task.

Stack Cup The task requires two stacking operations in a fixed order within one episode: the left blue cup is stacked on the yellow cup in the middle, and then the right blue cup is stacked on the same yellow cup. Initial cup positions are varied across trials. We evaluate each model over 32 trials. Each successful stack is worth 0.5 points, so a trial that completes only the first stack scores 0.5. We use the instruction "Stack the left blue cup on the yellow cup in the middle, then stack the right blue cup on the yellow cup in the middle." for this task.

Air Fryer This task requires sequential multi-stage manipulation with a Franka arm with a gripper: opening the air fryer basket, picking up the food item from the plate and placing it inside the basket, and closing the basket. We evaluate each model over 32 trials. We use the instruction "Open the air fryer basket, put the food from the plate into the basket, and close the basket." for this task.

Table Sweeping This bimanual task requires the two arms to play asymmetric and simultaneous roles: the left hand holds the dustpan steady against the floor while the right hand sweeps debris into it with a brush. The debris layout is varied across trials. We evaluate each model over 32 trials. A trial receives 0.5 points for grasping the dustpan correctly and another 0.5 points for sweeping the debris into it. We use the instruction "Hold the dustpan with the left hand and sweep the debris into it with the brush in the right hand" for this task.

Pump and Scrub This bimanual task requires picking up a scrubber with the left hand and scrubbing the interior of a white bowl, i.e., maintaining sustained periodic contact with a curved surface rather than executing a single grasp-and-place. We evaluate each model over 32 trials. A trial receives 0.5 points for pumping the soap and 0.5 points for scrubbing the bowl. We use the instruction "Pick up the blue scrubber with the left hand and scrub the white bowl" for this task.

Conveyor Package Handling The task requires picking up a parcel from the conveyor belt and placing it on the unloading table. Parcels are fed onto the belt by a human operator, so the robot has to handle each one as it arrives. To prevent memorization of one grasp pose, we vary the parcel across three categories that differ in size, rigidity, and the required grasp type, and vary the parcel position and orientation along the belt to evaluate spatial generalization. We evaluate each model over 5 trials, and each trial consists 10 consecutive packages. We use the instruction "Bring the package over, orient barcode up, then place it on the conveyor." for this task.

C.2 Hardware Setup

We evaluate on three platforms: OpenArm, ALLEX, and Franka Research 3, as shown in Figure 5.

OpenArm. OpenArm is a bimanual platform with two 7-DoF arms, each equipped with a 6-DoF Inspire RH56F1 hand. A ZED stereo camera is mounted on the neck, and its left and right views are used as visual inputs. We collect demonstrations by controlling the arms with ExoArm-7, a wearable exoskeleton for OpenArm, and the hands with Manus Pro gloves, where finger joint angles are computed from the fingertip positions via inverse kinematics. The neck is fixed during data collection.

ALLEX. ALLEX is a high-DoF upper-body humanoid. We use 48 joints: two 7-DoF arms, two 15-DoF five-finger hands, a 2-DoF waist, and a 2-DoF neck. A head-mounted stereo camera is the only visual input, with no wrist or side cameras. We collect demonstrations using a multi-device teleoperation system with separate interfaces for different robot components. A Meta Quest VR device controls the head and waist, while Vive Trackers provide wrist poses from which arm joint angles are obtained via inverse kinematics. Finger joint angles are similarly recovered from fingertip positions captured by Manus Pro gloves.

Franka Research 3. Following DROID (Khazatsky et al., 2024), we use a 7-DoF Franka Research 3 arm with a Robotiq 2F-85 gripper. An external ZED camera and a wrist-mounted ZED camera are used as visual inputs. We collect demonstrations via VR teleoperation with a Meta Quest controller.

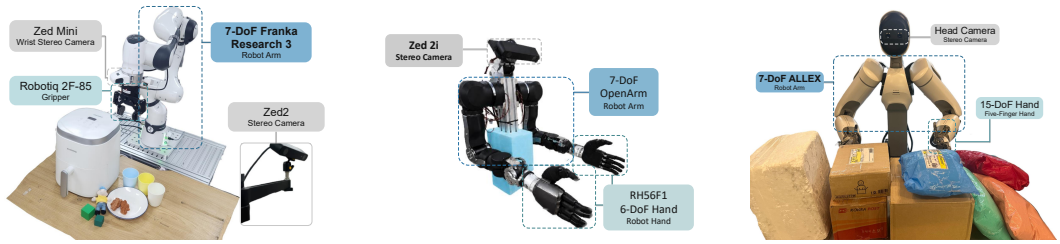


Figure 5: Hardware setups and task environments.

C.3 Simulation Hyperparameter Details

Training. We implement PaceVLA on two pretrained VLAs, GR00T N1.5 and RLDX-1-IMG, following their official implementations. For both models, we freeze the VLM backbone and train the action expert, the mode-specific decoders, the router, and the judge. We use $K{=}4$ temporal modes: two fine-grained modes $(16, 16)$ and $(8, 8)$, and two quantized modes whose configuration depends on the benchmark (Table 8). We set $\lambda_{\text{sup}}{=}0.1$, $\lambda_{\text{bal}}{=}0.05$, and $\lambda_{\text{conf}}{=}0.1$, where λ_{sup} and λ_{bal} increase linearly from zero over the first 5k iterations. In inference, the quantized group is selected when the judge's predicted quantizability score is at least the threshold δ. The remaining hyperparameters are summarized in Table 7.

Controller configuration. Both RoboCasa-Kitchen and LIBERO use an operational space controller (OSC). Since temporal action quantization merges consecutive delta actions into a single command, the controller must track a larger displacement per step, and the default position gain ($k_p{=}150$) can undershoot large merged motions. We therefore select $k_p \in \{150, 300, 600\}$ per task, and apply the same per-task gain to all acceleration methods for a fair comparison. For base GR00T N1.5, we use the default gain.

Table 7: Hyperparameter details for training PACEVLA on simulation benchmarks.

	GR00T N1.5 + PACEVLA	RLDX-1-IMG + PACEVLA
Optimizer	AdamW	AdamW
Optimizer momentum	$\beta_1, \beta_2 = 0.95, 0.999$	$\beta_1, \beta_2 = 0.9, 0.999$
Optimizer weight decay	1e-5	1e-5
Learning rate	1e-4	1e-4
Learning rate scheduler	Cosine decay	Cosine decay
Warmup iterations	3,000	3,000
Training iterations	60,000	60,000
$\lambda_{\text{sup}}, \lambda_{\text{bal}}, \lambda_{\text{conf}}$	0.1, 0.05, 0.1	0.1, 0.05, 0.1
Router weight w temperature	0.5	0.5
Target distribution r temperature	0.3	0.3
Minimum mode probability	0.05	0.05

Table 8: Benchmark-specific settings. Quantized modes are given as (H_k, T_k).

	RoboCasa-Kitchen	LIBERO	DexJoCo
Batch size	64	32	64
Quantized modes	$(16, 8),\ (8, 4)$	$(16, 9),\ (8, 5)$	$(16, 8),\ (8, 4)$

C.4 REAL HYPERPARAMETER DETAILS

Training. All real-world experiments use GR00T N1.5 + PACEVLA with the same training hyperparameters as in Table 7, except for the embodiment-specific settings in Table 9. For ALLEX, we use a longer source horizon, so the fine-grained modes are $(24, 24)$ and $(12, 12)$; the other embodiments use $(16, 16)$ and $(8, 8)$. On Franka, Pick-and-Place is trained for 20k iterations, while Air Fryer and Stack Cup are trained for 30k iterations.

Controller configuration. On the real robots, temporally quantized commands require both the arm and the end-effector to follow larger changes within a single control step. For the Franka arm, we use Cartesian impedance control with a translational stiffness of 600 N/m (800 N/m for stacking), and set the Robotiq 2F-85 to its maximum finger speed (150 mm/s) and minimum grasp force ($\sim$20 N). For OpenArm, we use joint PD control with gravity compensation, with shoulder and elbow gains of $K_p \in \{240, 480\}$ Nm/rad across two configurations.

Table 9: Embodiment-specific settings for real-world tasks. Modes are given as (H_k, T_k).

	Franka	OpenArm	ALLEX
Batch size	64	64	256
Training iterations	30,000	40,000	30,000
Fine-grained modes	$(16, 16),\ (8, 8)$	$(16, 16),\ (8, 8)$	$(24, 24),\ (12, 12)$
Quantized modes	$(16, 8),\ (8, 4)$	$(16, 8),\ (8, 4)$	$(24, 12),\ (12, 6)$

C.5 QUANTIZABILITY SUPERVISION DETAILS

We design the annotation criteria with the assistance of Claude Opus 5 (Anthropic, 2026). We use per-task success rates under fixed quantization only as task-level cues for constructing the quantizability criteria, and they are never used as supervision targets for the judge. These task-level cues help reveal quantizability patterns that are difficult to infer from task information alone and guide the construction of the visual checks. The criterion is decomposed into four to six concrete visual checks. Each check is rated on a 1–5 scale and contributing either positively or negatively to quantizability (prompts in Appendix E).

We then annotate the full datasets with Gemini 3.8 Flash (Google DeepMind, 2026). For each action chunk, the model receives the first-frame camera observations, task instruction, and computed facts

derived from the demonstration, and returns one integer rating per check. The ratings are combined with fixed weights and normalized to $[0, 1]$ to obtain the quantizability score y.

C.6 Comparison to SAIL

We reimplement SAIL (Arachchige et al., 2025) on the same GR00T-N1.5 and RLDX-1-IMG backbones used in our experiments. We follow the original implementation and hyperparameters, with three changes to keep the comparison fair. First, we parameterize action chunks as end-effector deltas to match the delta control interface of simulation benchmarks. Second, for Error-Adaptive Guidance (EAG), we condition on the four actions of the previous chunk that immediately follow the replanning step, *i.e.*, actions replaced by the new chunk, so that the condition is temporally aligned with the prediction. Third, we disable action scheduling. Instead, SAIL follows the same inference setting as our method and the baselines: synchronous inference on all tasks except ALLEX conveyor task, where all methods use asynchronous inference with real-time chunking (RTC) (Black et al., 2026). Thus, speedups within each task come from faster motion rather than from the inference setting.

Table 10: **Quantized-action prediction accuracy on RoboCasa-Kitchen.** *Ours* uses dedicated coarse decoders, while *Base* quantizes GR00T N1.5 predictions using the same SO(3)-aware rule. (H, q) denotes the source horizon and quantization level, and *Both* averages the two configurations.

Decoder (H, q)	Metric	*Ours*	*Base*
(16,2)	MSE ↓	**0.0273**	0.0344
	DTW ↓	**3.209**	3.777
	Cosine ↑	**0.850**	0.809
(8,2)	MSE ↓	**0.0163**	0.0243
	DTW ↓	**1.359**	1.658
	Cosine ↑	**0.886**	0.847
Both	MSE ↓	**0.0218**	0.0294
	DTW ↓	**2.284**	2.718
	Cosine ↑	**0.868**	0.828

D Additional Analysis

D.1 Quantitative Analysis on Ground-Truth Data

Setup. We examine whether dedicated coarse decoders provide an advantage over directly quantizing the actions predicted by a standard policy. We evaluate on the RoboCasa-Kitchen simulation benchmark (Nasiriany et al., 2024), sampling trajectory points across all 24 tasks. Both PaceVLA and the baselines are fine-tuned on the same data and receive the same ground-truth observations. The two predictions are constructed as follows:

- **Ours**: The output of the quantized decoder D_k.
- **Base**: The output of decoder, post-processed using the same quantization rule as the target of D_k.

Both are compared against the corresponding ground-truth quantized target $\mathbf{a}^{(k)}$.

Metrics. Let $\hat{\mathbf{a}} = (\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_T)$ and $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_T)$ denote the predicted and ground-truth quantized chunks, respectively, with $\mathbf{a}_i \in \mathbb{R}^d$ over the continuous action dimensions. We report MSE, cosine similarity, and Dynamic Time Warping (DTW).

MSE measures pointwise magnitude error:

$$\mathrm{MSE}(\hat{\mathbf{a}}, \mathbf{a}) = \frac{1}{Td} \sum_{i=1}^{T} \|\hat{\mathbf{a}}_i - \mathbf{a}_i\|_2^2 . \tag{9}$$

Cosine similarity measures directional agreement:

$$\cos(\hat{\mathbf{a}}, \mathbf{a}) = \frac{\mathrm{vec}(\hat{\mathbf{a}})^\top \mathrm{vec}(\mathbf{a})}{\|\mathrm{vec}(\hat{\mathbf{a}})\|_2 \, \|\mathrm{vec}(\mathbf{a})\|_2}. \tag{10}$$

DTW measures sequence similarity while allowing temporal misalignment. With $D(0,0) = 0$ and $D(i,0) = D(0,j) = \infty$ for $i, j \geq 1$,

$$\begin{aligned} D(i,j) = \|\hat{\mathbf{a}}_i - \mathbf{a}_j\|_2 \\ + \min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\} , \end{aligned} \tag{11}$$

and $\mathrm{DTW}(\hat{\mathbf{a}}, \mathbf{a}) = D(T,T)$.

Results. Table 10 shows that the dedicated coarse decoders consistently outperform post-hoc quantization of the baseline predictions across both configurations and all three metrics. This indicates that directly learning the quantized action target better preserves the ground-truth motion than quantizing a fine-grained prediction after inference.

D.2 Rollout Trajectory Comparison

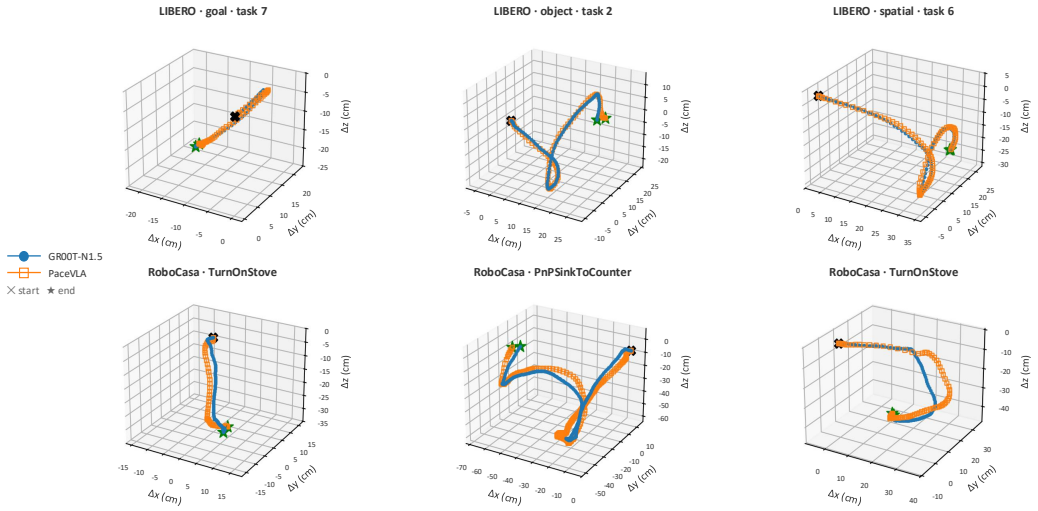


Figure 6: **End-effector rollout trajectories.** End-effector trajectories of the base policy (blue) and PaceVLA (orange) from the same initial state; PaceVLA traverses the same path with fewer, sparser steps.

D.3 Failure Case Analysis

Across the failed rollouts we inspected (Figure 7), most failures seem to come from perception rather than from the temporal decision. All four decoders use the same visual and language features from the backbone, so if the scene is misread, the choice of decoder makes little difference. In Pick-and-Place, most failures happen during grasping. The judge assigns low quantizability in this phase and the router switches to fine-grained modes, but the trajectory is aimed slightly off the object, so the gripper knocks the cup over and closes on air, or holds the object loosely and drops it during transport. In Stack Cup, the gripper misses the first cup slightly and hits its edge, so the first stack fails and the robot cannot recover. The base model fails in the same way, so we believe these errors are mostly due to perception rather than temporal adaptation.

We also see a similar pattern in the OpenArm bimanual hand tasks. The only visual input is an egocentric stereo camera, and the hands and tools often block the view of the contact region. In Table Sweeping, the left hand grasps the dustpan at the wrong angle, so its lip does not sit flush against the table, and the debris swept by the right hand piles up against the lip instead of going in. In Pump and Scrub, the policy pumps soap onto the scrubber but never proceeds to scrubbing, possibly because the hands occlude the scrubber from the camera, so the policy cannot see that the soap has already been applied. In these cases the missing information is in the observation, not in the trajectory, so finer temporal granularity is unlikely to help. These cases suggest that PACEVLA is still limited by the quality of its perception, and that better sensing, such as depth or tactile input, would add to the gains from adaptive temporal execution.

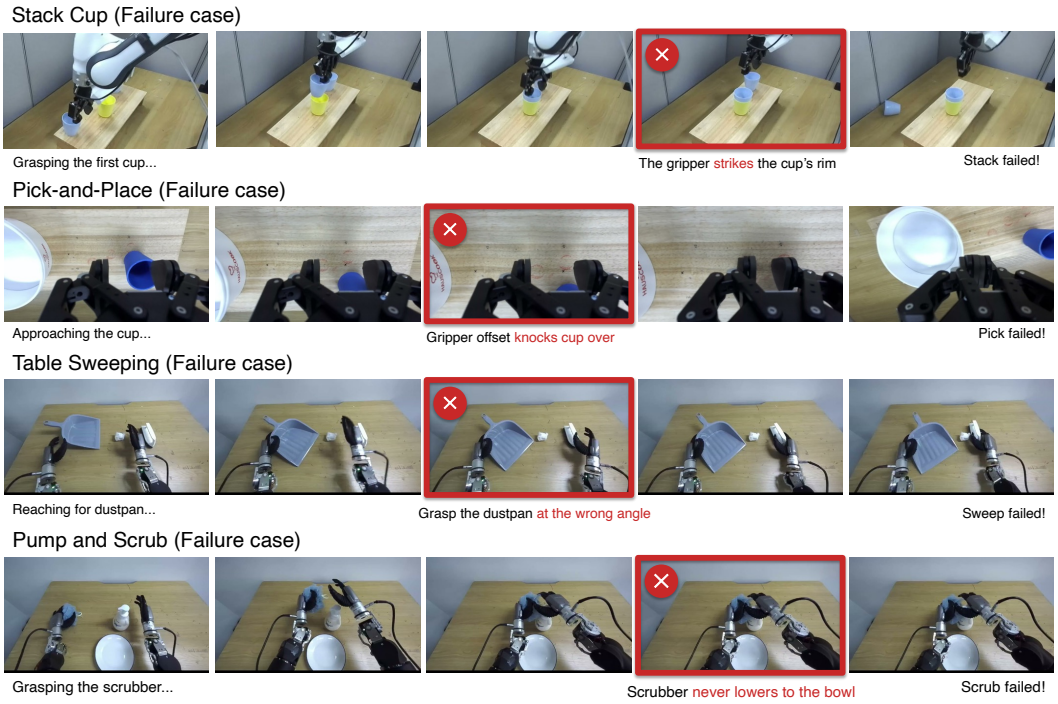


Figure 7: Failure cases of PACEVLA. Most failures come from small targeting errors (Pick-and-Place, Stack Cup) or from the hands and tools blocking the contact region in the egocentric view (Table Sweeping, Pump and Scrub).

E PROMPTS USED ON DATASET ANNOTATION

Sim: RoboCasa-Kitchen

```
[USER]
<3 images>
You are judging one moment of a kitchen robot's motion, to decide whether the arm could
↪  get through the stretch of motion ahead of it faster -- reaching the same places in
↪  fewer and longer steps -- without changing the outcome.

Going faster leaves the arm fewer chances to correct itself along the way. That costs most
↪  where the arm still has to arrive somewhere exactly: setting down what it is carrying
↪  onto the spot it must end up on, or closing the gripper on one exact point -- and,
↪  when it is the gripper closing, more still if that spot is hemmed in. It costs least
↪  where the arm only acts on something that is anchored in place -- a button or switch
↪  pressed, a knob or tap turned, a door pushed shut. There a longer step just presses or
↪  turns a little harder and nothing is lost.

Judge the moment in front of you, not the task as a whole. One task passes through both
↪  kinds from one moment to the next.

You are shown 3 camera views: agentview-left, agentview-right, and a wrist (eye-in-hand)
↪  close-up. The wrist camera is mounted on the gripper, so objects normally look close
↪  in it -- general closeness is normal. Use the wrist view to judge the close moments:
↪  the gripper actually closing on something, and something being set down onto the spot
↪  it must end up on.

The robot was told: {instruction}

{computed facts}

The measurements above are stated as fact -- do not re-estimate or repeat them. Answer
↪  each check from what the cameras show about the MOMENT in front of you, read together
↪  with those measurements.
Answer each check on its own line as "A) 3", in order, nothing else -- one digit from 1 to
↪  5 per check, rating how far that check describes this moment:
  5 = clearly true of this moment
  4 = mostly true
  3 = partly true
  2 = barely true
  1 = not true at all
A grade refers only to the check on that line.
A) Is the gripper CLOSING on its target right now, or about to -- the fingers
   already around the handle, rim or object and only the squeeze left, so that a
   slip here loses it?
B) Is that closing spot hemmed in -- reached into a cabinet, a microwave, an oven or
   a sink, or lined up with a slot -- so the gripper has little room around it as
   it closes?
C) Is the arm just pressing, turning or pushing something that is anchored in
   place -- a button or switch, a knob or tap, a door or drawer being shoved shut
   -- and not something it has to keep hold of while it travels, the way a handle
   is held on to and pulled to swing a door or drawer open?
D) Is the object ALREADY RESTING where it is meant to end up, taking its own
   weight, with nothing left but to open the hand and draw back?
E) Is the gripper carrying something through open space -- already holding it,
   still well away from wherever it is going?
F) Is the arm STILL LOWERING the object onto its target right now -- the gripper
   not yet opened, the object not yet taking its own weight, coming down onto
   the spot it must end up on?
Answer:
```

Sim: LIBERO

```
[USER]
<2 images>
You are judging one moment of a table-top robot's motion, to decide whether the arm could
↪  get through the stretch of motion ahead of it faster -- reaching the same places in
↪  fewer and longer steps -- without changing the outcome.

Going faster leaves the arm fewer chances to correct itself along the way. That costs most
↪  when the gripper has to close on one exact thing -- and more still when the way in is
↪  threaded: the object perched on top of another, or wedged in among things standing
↪  close by, so that there is one line in and a longer step swings wider than that line.
↪  It costs least where whatever is being moved only has to land somewhere that will
↪  catch it, and at the moments either side of the work -- crossing open space, carrying
↪  something with room all round it, drawing back after letting go.

Judge the moment in front of you, not the task as a whole. One task passes through both
↪  kinds from one moment to the next: the same episode that threads a bowl out from
↪  between two dishes then carries it through open air.

You are shown 2 camera views of this one moment: a scene view and a wrist (eye-in-hand)
↪  close-up. The wrist camera is mounted on the gripper, so objects normally look close
↪  in it -- general closeness is normal. Use the wrist view to judge the close moments:
↪  the gripper actually closing on something, and something being set down into the place
↪  it must end up.

The robot was told: {instruction}

{computed facts}

The measurements above are stated as fact -- do not re-estimate or repeat them. Answer
↪  each check from what the cameras show about the MOMENT in front of you, read together
↪  with those measurements.
Answer each check on its own line as "A) 3", in order, nothing else -- one digit from 1 to
↪  5 per check, rating how far that check describes this moment:
  5 = clearly true of this moment
  4 = mostly true
  3 = partly true
  2 = barely true
  1 = not true at all
A grade refers only to the check on that line.
A) Is the place this object has to end up ROOMY FOR IT -- a wide bowl, an open
   basket, a broad tray or an open stretch of table with slack all round -- rather
   than a plate, a rack, or a spot barely bigger than the thing itself?
B) Is the gripper closing on the object right now, or about to -- the fingers
   already around it and only the squeeze left, so that a slip here loses it?
C) Is that closing spot threaded -- the thing being taken perched on top of another
   object, or wedged in among things standing close by -- so that there is only one
   line in for the hand?
D) Is the hand ALREADY OPEN and coming away, the object left sitting where it was
   meant to end up and holding its own weight without the fingers?
E) Is this a job that needs no hold kept on anything -- a push, a knob turned, a
   drawer pressed shut -- so that once it is sent it carries on without the hand?
Answer:
```

Sim: DexJoCo

```
[USER]
[IMAGE]
[IMAGE]
You are judging one moment of a robot's motion with a dexterous hand, to decide whether
↪  the arm could get through the stretch of motion ahead of it faster -- reaching the
↪  same places in fewer and longer steps -- without changing the outcome.

Going faster leaves the arm fewer chances to correct itself along the way. Changing the
↪  working configuration of a held object and directing a held tool onto a localized
↪  target depend on the relation between the fingers, object and target. Maintaining a
↪  supported press and carrying an already grasped object through clear space impose
↪  different constraints. Use the visible interaction to judge these checks; do not infer
↪  contact or a secure grasp from commanded motion alone.

Judge the moment in front of you, not the task as a whole. One task passes through
↪  different interactions from one moment to the next.

You are shown 2 camera views of this one moment: image 1 is the scene view
↪  (observation.images.ego_right), and image 2 is a wrist (eye-in-hand) close-up
↪  (observation.images.wrist). The wrist camera is mounted on the hand, so objects
↪  normally look close in it -- general closeness is normal. Use the wrist view to judge
↪  the close moments: the fingers operating the held object, a tool approaching its
↪  target, and a finger pressing a supported target.

The robot was told: {instruction}

{computed facts}

The measurements above are stated as fact -- do not re-estimate or repeat them. Answer
↪  each check from what the cameras show about the MOMENT in front of you, read together
↪  with those measurements.
Answer each check on its own line as "A) 3", in order, nothing else -- one digit from 1 to
↪  5 per check, rating how far that check describes this moment:
  5 = clearly true of this moment
  4 = mostly true
  3 = partly true
  2 = barely true
  1 = not true at all
A grade refers only to the check on that line.
A) Are the fingers operating movable parts of the object being held|closing
   jaws or folding an articulated part|so that changing the relation between
   those parts is the operation, rather than merely holding the object?
B) Is a held tool being directed into an impact on a localized target, rather
   than making a sustained supported press? Use visible held-tool/target geometry together
   ↪  with the command motion to judge
   preparation for or execution of a targeted strike. Command targets do not establish
   that an impact actually occurred; the presence of a hammer alone is not enough.
C) Is a finger or hand pressing a target supported by the surrounding surface,
   without lifting it, striking it with a tool, or manipulating movable parts
   within a grasp? A surface supports the target; do not assume it is bolted down.
D) Is an already grasped object being moved or tilted as a whole through clear
   space, without changing the working configuration of its movable parts,
   acquiring/releasing the grasp, or entering a visibly tight target region?

Answer:
```

Real-World: Franka+Gripper Tasks

```
[USER]
<2 images>
You are judging one moment of a table-top robot's motion, to decide whether the arm could
↪  get through the stretch of motion ahead of it faster -- reaching the same places in
↪  fewer and longer steps -- without changing the outcome.

Going faster leaves the arm fewer chances to correct itself along the way. That costs most
↪  at the two moments where the object can be lost: when the hand is closing on it, and
↪  when it is being brought down into the place it has to end up. Both get worse the less
↪  room there is -- letting a cube down into a cup barely wider than itself leaves no
↪  slack, while letting the same cube down onto an open plate does.

Most moments are neither. Crossing empty space to reach for something, carrying something
↪  with room all round it, or drawing the hand back after letting go -- a longer step
↪  changes nothing there. That is the common case.

Judge the moment in front of you, not the task as a whole. One task passes through both
↪  kinds from one moment to the next.

You are shown 2 camera views of this one moment: a scene view and a wrist (eye-in-hand)
↪  close-up. The wrist camera is mounted on the gripper, so objects normally look close
↪  in it -- general closeness is normal. Use the wrist view to judge the close moments:
↪  the hand actually closing on the object, and the object being brought down into the
↪  place it must end up.

The robot was told: {instruction}

{computed facts}

The measurements above are stated as fact -- do not re-estimate or repeat them. Answer
↪  each check from what the cameras show about the MOMENT in front of you, read together
↪  with those measurements.
Answer each check on its own line as "A) 3", in order, nothing else -- one digit from 1 to
↪  5 per check, rating how far that check describes this moment:
  5 = clearly true of this moment
  4 = mostly true
  3 = partly true
  2 = barely true
  1 = not true at all
A grade refers only to the check on that line.
A) Is the hand closing on the object right now, or about to -- the fingers
   already around it and only the squeeze left, so that a slip here loses it?
B) Is the place this object has to end up ROOMY FOR IT -- a broad plate or an
   open dish with slack all round -- rather than a cup or a container barely
   wider than the thing itself?
C) Is the hand ALREADY OPEN and coming away, the object left sitting where it
   was meant to end up and holding its own weight without the fingers?
D) Is the arm STILL LOWERING the object onto its target right now -- the hand
   not yet opened, the object not yet taking its own weight, coming down onto
   the spot it must end up on?
Answer:
```

Real-World: OpenArm+Bimanual Hand Tasks

```
[USER]
<2 images>
You are judging one moment of a two-armed robot's motion, each arm ending in a dexterous
↪  hand, to decide whether it could get through the stretch of motion ahead faster --
↪  reaching the same places in fewer and longer steps -- without changing the outcome.

Going faster leaves the arms fewer chances to correct themselves along the way. That costs
↪  most where a hand has to arrive somewhere exactly or close on something exactly:
↪  taking hold of a tool, pushing a hand-held thing down onto something that is only
↪  standing on the table and can tip or slide, and guiding what is held into a small
↪  target region. It costs least where a hand is crossing open space, carrying something
↪  that is already firmly in the grasp with room all round it.

The two hands may be doing different things at the same moment -- one may hold something
↪  steady while the other works. Read both. Use the visible interaction; do not infer
↪  contact or a secure grasp from commanded motion alone.

Judge the moment in front of you, not the task as a whole. One task passes through
↪  different interactions from one moment to the next.

You are shown 2 camera views of this one moment, both scene views of the same workspace
↪  from slightly different angles (image 1 is observation.images.ego_left, image 2 is
↪  observation.images.ego_right). There is no wrist camera; neither view is an
↪  eye-in-hand close-up. This robot has two arms, each with a dexterous hand, and they
↪  may be doing different things at the same time -- read both hands.

The robot was told: {instruction}

{computed facts}

The measurements above are stated as fact -- do not re-estimate or repeat them. Answer
↪  each check from what the cameras show about the MOMENT in front of you, read together
↪  with those measurements.
Answer each check on its own line as "A) 3", in order, nothing else -- one digit from 1 to
↪  5 per check, rating how far that check describes this moment:
  5 = clearly true of this moment
  4 = mostly true
  3 = partly true
  2 = barely true
  1 = not true at all
A grade refers only to the check on that line.
A) Is a hand closing on a tool or object to take hold of it right now, or about
   to -- the fingers already around it and only the squeeze left, so that a slip
   here loses it -- rather than already holding it securely?
B) Is a hand pushing down on something that is only standing on the table and is
   not held flat by it -- a bottle, a pump head, an upright container -- so that
   going too far tips it over or slides off it, rather than pressing on something
   the table holds down?
C) Is what a hand holds being guided into or over a small target region -- across
   a rim or lip, into a narrow opening, onto a spot only a little larger than the
   thing itself -- so that missing that line loses the attempt?
D) Is an already grasped object being moved or tilted as a whole through clear
   space, without changing the working configuration of its movable parts,
   acquiring/releasing the grasp, or entering a visibly tight target region?
E) Is a hand moving through open space touching nothing at all -- on its way
   somewhere, or drawing back after letting go?

Answer:
```

Real-World: Allex Conveyer Package Handling Task

```
[USER]
<2 images>
You are judging one instant of a two-armed robot with hands, to decide how much of the
↪  next stretch of its motion could be thinned out -- how many of its commanded poses
↪  could be dropped, letting the arms travel further between the ones that remain,
↪  without changing the outcome.

What thinning takes away is the arms' chance to correct on the way, and there are two ways
↪  that loses the object. One is a hold that exists only as a balance: the thing is
↪  trapped between two hands and nothing is closed around it, so the hold IS the push of
↪  one hand against the other. If that thing is stiff it cannot give, and a coarser swing
↪  changes the balance and drops it. Something soft squashes and stays pinched.

The other is a load that does not hold its own shape. A bag or a sagging thing hangs,
↪  swings and shifts while it is carried, and the further the arms travel between
↪  commanded poses the further it swings.

Most moments are neither. Sliding something across to the other side, pushing something
↪  that is still resting on a surface, or moving with nothing in the hands -- a longer
↪  step changes nothing.

Judge the moment in front of you, not the name of the job. One segment passes through
↪  several of these from one second to the next.

The two images are the left and right camera views of this one moment.

The robot was told: {instruction}

{computed facts}

The measurements above are stated as fact -- do not re-estimate or repeat them. Answer
↪  each check from what the cameras show about the MOMENT in front of you, read together
↪  with those measurements.
Answer each check on its own line as "A) 3", in order, nothing else -- one digit from 1 to
↪  5 per check, rating how far that check describes this moment:
  5 = clearly true of this moment
  4 = mostly true
  3 = partly true
  2 = barely true
  1 = not true at all
A grade refers only to the check on that line.
A) Are the two hands SQUEEZING something between them to hold it, and is that
   thing HARD -- a box, a carton, something that will not squash?
B) Is the item a POUCH OR PACKET -- a soft mailer, a padded envelope, a
   plastic parcel -- rather than a rigid carton with square sides?
C) Is the thing only being SENT ACROSS -- slid or passed over to the other side,
   or shoved on its way -- with no particular spot it has to come to rest on?
D) Are the hands ON THEIR WAY -- reaching out towards something, or drawing back
   from what they have finished, carrying nothing between them?
Answer:
```