PriorVLA: Prior-Preserving Adaptation for Vision-Language-Action Models

Xinyu Guo[1,3,4,†] **Bin Xie**[2,‡] **Wei Chai**[5,†] **Xianchi Deng**[2]
Tiancai Wang[2] **Zhengxing Wu**[1✉] **Xingyu Chen**[4✉]

[1]Institute of Automation, Chinese Academy of Sciences [2]Dexmal
[3]University of Chinese Academy of Sciences [4]Zhongguancun Academy
[5]Nanjing University of Aeronautics and Astronautics

Abstract

Large-scale pretraining has made Vision-Language-Action (VLA) models promising foundations for generalist robot manipulation, yet adapting them to downstream tasks remains necessary. However, the common practice of full fine-tuning treats pretraining as initialization and can shift broad priors toward narrow training-distribution patterns. We propose PriorVLA, a novel framework that preserves pretrained priors and learns to leverage them for effective adaptation. PriorVLA keeps a frozen Prior Expert as a read-only prior source and trains an Adaptation Expert for downstream specialization. Expert Queries capture scene priors from the pretrained VLM and motor priors from the Prior Expert, integrating both into the Adaptation Expert to guide adaptation. Together, PriorVLA updates only 25% of the parameters updated by full fine-tuning. Across RoboTwin 2.0, LIBERO, and real-world tasks, PriorVLA achieves stronger overall performance than full fine-tuning and state-of-the-art VLA baselines, with the largest gains under out-of-distribution (OOD) and few-shot settings. PriorVLA improves over $\pi_{0.5}$ by 11 points on RoboTwin 2.0-Hard and achieves 99.1% average success on LIBERO. Across eight real-world tasks and two embodiments, PriorVLA reaches 81% in-distribution (ID) and 57% OOD success with standard data. With only 10 demonstrations per task, PriorVLA reaches 48% ID and 32% OOD success, surpassing $\pi_{0.5}$ by 24 and 22 points, respectively. Project Page: https://priorvla.github.io/

1 Introduction

Vision-Language-Action (VLA) models have emerged as promising foundations for generalist robot manipulation [1, 2, 3, 4, 5]. Through large-scale pretraining on diverse robot data, a single policy can learn broad priors that support general manipulation capabilities, grounding language instructions in visual observations and generating actions across tasks and embodiments [6, 7, 8, 9, 10, 11]. Yet the complexity and diversity of real-world manipulation cannot be fully covered by pretraining alone, making downstream adaptation necessary [12, 13, 14, 15, 16, 17]. In practice, downstream adaptation is commonly performed via full fine-tuning, which can quickly improve in-distribution (ID) performance [12, 5]. However, full fine-tuning treats the pretrained model mainly as an initialization and updates all parameters to fit limited task-specific data, which can shift broad pretrained priors toward narrow training-distribution patterns and weaken out-of-distribution (OOD) generalization [5, 12, 18, 19, 16]. Under this paradigm, generalizable adaptation requires additional diverse demonstrations to cover broader variations [20, 21, 22, 23], which is a poor use of large-scale pretraining.

Large-scale VLA pretraining is valuable not only as an effective initialization for downstream fine-tuning, but also as a source of broad priors for generating robot actions from visual observations and

[†] Work done during an internship at Dexmal. [‡] Project leader. [✉] Corresponding author.

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

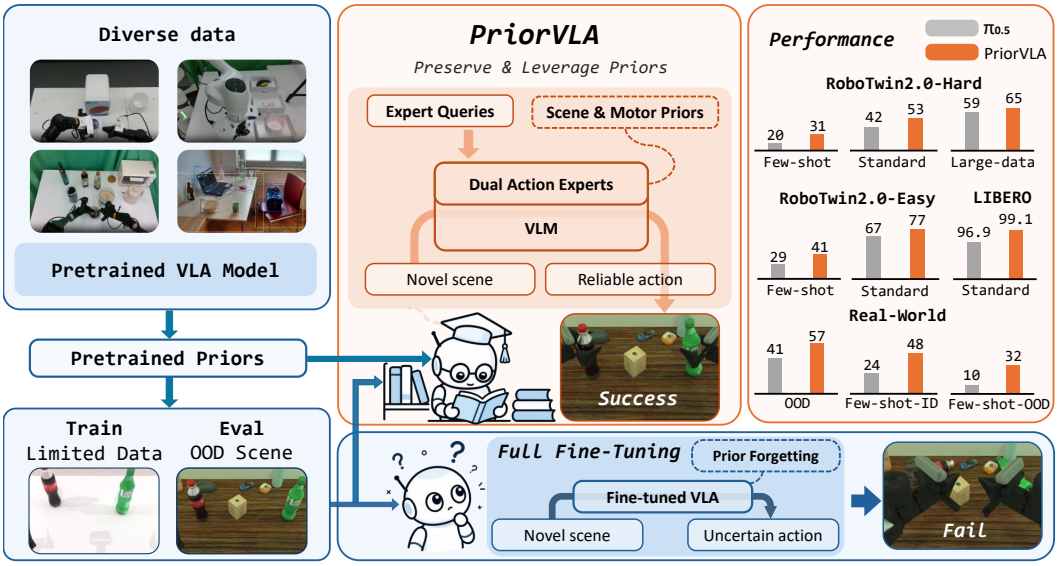


Figure 1: **Overview of PriorVLA.** Large-scale pretraining provides broad priors for general manipulation, but full fine-tuning on limited downstream data can treat these priors mainly as initialization and lead to prior forgetting, especially when evaluated under OOD scenes. PriorVLA instead preserves and leverages pretrained scene and motor priors through Dual Action Experts and Expert Queries, improving adaptation across simulation and real-world tasks with the largest gains in few-shot and OOD settings.

language instructions across tasks and embodiments. Effective VLA adaptation should therefore preserve these priors and make them usable for downstream tasks. These priors are not explicit rules or final action outputs, but emerge in the pretrained model's forward pass. They provide complementary scene and motor priors: the VLM encodes task-relevant visual structure, while the action expert encodes action-generation regularities. These priors supplement limited downstream data with pretrained structure, helping the adapted policy handle novel observations and generate reliable actions in few-shot and OOD settings. Existing adaptation methods either reduce or constrain parameter updates [24, 12, 18, 19], or bridge frozen vision-language features to action policies [16, 17]. However, they primarily focus on updating, restricting, or connecting pretrained components, rather than explicitly preserving pretrained forward-pass representations as usable prior sources. What remains missing is a mechanism that preserves these priors and provides learnable interfaces for the adapted policy to use them.

To this end, we propose **PriorVLA**, a prior-preserving adaptation framework that preserves pretrained priors and learns to leverage them during downstream adaptation, as summarized in Fig. 1. PriorVLA introduces Dual Action Experts to separate prior preservation from task specialization. The original pretrained action expert is kept as a frozen Prior Expert, while a trainable Adaptation Expert, initialized from the same weights, specializes to the downstream task. The Prior Expert is not used for its final action prediction; instead, it serves as a read-only forward path whose internal representations provide motor priors from pretraining. PriorVLA further introduces Expert Queries as learnable interfaces for using preserved priors. Scene Queries capture task-relevant scene priors from the pretrained VLM, Motor Queries capture motor priors from the Prior Expert, and Action Queries integrate these priors inside the Adaptation Expert to guide action generation. Although trained with the same action objective as full fine-tuning, PriorVLA changes what adaptation learns: it learns to capture and integrate preserved priors, rather than adapting by shifting them toward narrow training-distribution patterns.

We conduct extensive evaluations of PriorVLA across simulation and real-world settings, including RoboTwin 2.0 [22], LIBERO [25], and eight real-world tasks spanning two embodiments: a single-arm Franka robot and a dual-arm AC-One robot platform. PriorVLA achieves stronger overall performance than full fine-tuning and state-of-the-art VLA baselines, with the largest gains under out-of-distribution evaluation and limited downstream supervision. On RoboTwin 2.0-Hard, PriorVLA

surpasses $\pi_{0.5}$ by 11 points, demonstrating stronger OOD generalization. On LIBERO, it achieves 99.1% average success across four task suites. Across eight real-world tasks, PriorVLA reaches 81% ID and 57% OOD success in the standard-data setting. With only 10 demonstrations per task, PriorVLA reaches 48% ID and 32% OOD success, surpassing $\pi_{0.5}$ by 24 and 22 points, respectively. These gains are achieved while updating only 25% as many parameters as full fine-tuning.

Our contributions are summarized as follows:

- We motivate a prior-preserving view of downstream VLA adaptation: beyond serving as initialization, large-scale pretraining provides forward-pass priors that should be preserved and leveraged during adaptation. This reframes adaptation as learning to use pretrained structure, rather than only fitting limited downstream data.
- We propose PriorVLA, a prior-preserving adaptation framework with two coupled designs: Dual Action Experts separate prior preservation from task specialization, and Expert Queries capture and integrate preserved scene and motor priors for action generation.
- We demonstrate stronger overall performance across RoboTwin 2.0, LIBERO, and eight real-world tasks on two embodiments, with the largest gains in OOD and few-shot settings, while updating only 25% as many parameters as full fine-tuning. Ablations further show that the preserved pretrained priors, rather than simply adding frozen branches or reducing trainable parameters, are critical to these gains.

2 Related Work

Pretrained VLA Models. Robotic manipulation policies trained on narrow task-specific datasets often struggle to generalize beyond their training distributions [26, 27, 28, 29, 30, 31]. Recent Vision-Language-Action (VLA) models address this limitation by scaling robot data, simulation experience, and pretrained vision-language backbones. Large-scale real-robot datasets and synthetic data generators have substantially expanded manipulation coverage across tasks, scenes, objects, and embodiments [1, 6, 7, 8, 32, 20, 21, 22, 23, 33]. Built on these data sources, RT-2, OpenVLA, and π_0 demonstrate increasingly generalist robot policies through large-scale pretraining [2, 3, 4], while recent systems further improve action modeling, data mixtures, and embodied reasoning [5, 34, 35, 36, 37]. These advances make pretrained VLAs valuable foundations for downstream manipulation, as many pretrained capabilities are difficult to recover from limited task-specific data.

Downstream Adaptation of VLA Models. Adapting pretrained VLAs to new tasks, robots, and data regimes is still challenging: full fine-tuning can quickly improve in-distribution performance, but may over-specialize the model and weaken pretrained capabilities. Recent work improves VLA adaptation by refining action representations and fine-tuning objectives [12], reducing trainable parameters with low-rank updates [24], or freezing/protecting the VLM while placing more adaptation capacity in action-side modules [35, 16]. Other approaches improve robustness by constraining or merging parameter updates [18, 19], incorporating temporal context [38], or preserving previous capabilities through continual-learning regularization and replay [39, 40]. Lightweight interface designs further connect vision-language representations to action policies through query-style tokens or bridge attention [12, 17]. RICL [41] and ICRT [42] use demonstration context for task adaptation, whereas PriorVLA learns an adapted policy without demonstration prompts at inference. How to preserve and leverage pretrained priors during downstream action generation remains less explored in current VLA adaptation frameworks.

3 Method

3.1 Problem Formulation

We consider a vision-language-action (VLA) policy that predicts a future action chunk for robot manipulation. Given visual observation o_t, language instruction l, and proprioceptive state s_t, the policy outputs an action chunk $\mathbf{a}_{t:t+H-1}$, where H denotes the action horizon.

A common pretrained VLA instantiation couples a vision-language model (VLM) with a flow-matching-based action expert (AE) [4, 5]. During denoising, the policy conditions on a noisy action

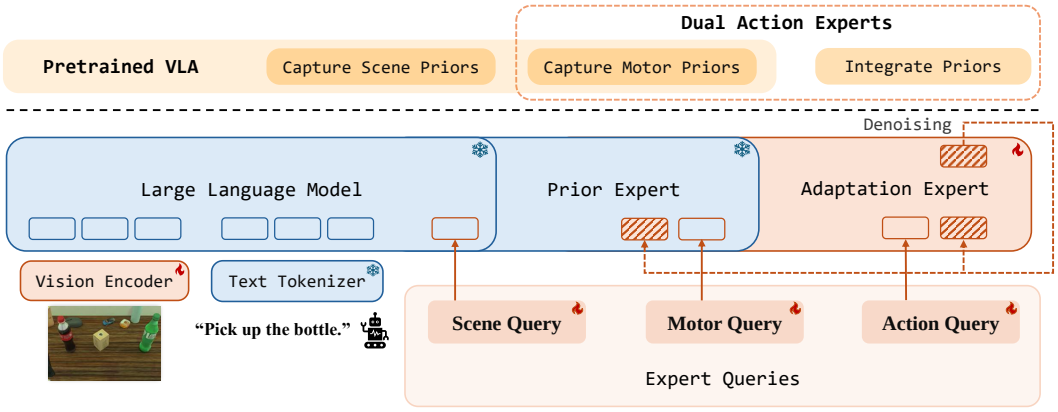


Figure 2: **PriorVLA architecture.** PriorVLA builds on a pretrained VLA and introduces two coupled modules: Dual Action Experts and Expert Queries. Dual Action Experts keep the original AE as a frozen Prior Expert and train an Adaptation Expert for downstream action generation. Expert Queries capture scene and motor priors from pretrained forward paths and integrate them into the Adaptation Expert; the Prior Expert serves only as a read-only prior source, while the Adaptation Expert drives denoising and produces the final action chunk.

chunk $\tilde{\mathbf{a}}_{t:t+H-1}$:

$$\hat{\mathbf{a}}_{t:t+H-1} = \pi_\theta(\tilde{\mathbf{a}}_{t:t+H-1},\, o_t,\, l,\, s_t)\,, \tag{1}$$

where $\hat{\mathbf{a}}_{t:t+H-1}$ denotes the denoising prediction. Through large-scale pretraining, the VLM acquires general vision-language representations, while the AE learns reusable motor priors for action generation. These complementary pretrained capabilities form the basis for downstream adaptation.

3.2 Overview

As shown in Fig. 2, PriorVLA instantiates prior-preserving adaptation with two coupled modules: *Dual Action Experts* (DAE) and *Expert Queries* (EQ). DAE separates prior preservation from task specialization by retaining the pretrained AE as a frozen *Prior Expert* and training an *Adaptation Expert* for downstream action generation. EQ provides learnable interfaces for using preserved priors: *Scene Queries* and *Motor Queries* capture scene and motor priors from pretrained forward paths, while *Action Queries* integrate these priors inside the *Adaptation Expert*. Only the *Adaptation Expert*'s denoising prediction is used for action generation.

3.3 Dual Action Experts

A pretrained AE contains reusable motor priors encoded in its denoising dynamics, but full fine-tuning can shift these priors toward narrow training-distribution action patterns. PriorVLA decouples prior preservation from downstream specialization by branching the pretrained AE into *Dual Action Experts*.

Specifically, we retain the original pretrained AE as a frozen *Prior Expert* and introduce a trainable *Adaptation Expert*, initialized from the same pretrained weights, for downstream action generation. During denoising, both experts are executed along the same noisy action trajectory. At denoising step τ, they receive the same noisy action chunk $\tilde{\mathbf{a}}^{\tau}_{t:t+H-1}$. The *Prior Expert* provides internal denoising representations as motor-prior features, while its denoising output is discarded. Only the denoising output of the *Adaptation Expert* updates the trajectory, which produces the final action chunk after denoising. For clarity, we omit the action-horizon subscript in the recurrence:

$$\tilde{\mathbf{a}}^{\tau}_{\mathrm{PE}} = \tilde{\mathbf{a}}^{\tau}_{\mathrm{Ada}} = \tilde{\mathbf{a}}^{\tau}, \qquad \tilde{\mathbf{a}}^{\tau+1} = \mathrm{FM}(\tilde{\mathbf{a}}^{\tau}, f^{\tau}_{\mathrm{Ada}})\,, \tag{2}$$

where f^{τ}_{Ada} denotes the denoising output of the *Adaptation Expert*, and $\mathrm{FM}(\cdot)$ denotes the flow-matching update. The updated chunk $\tilde{\mathbf{a}}^{\tau+1}$ is then used as the input to both experts at the next denoising step.

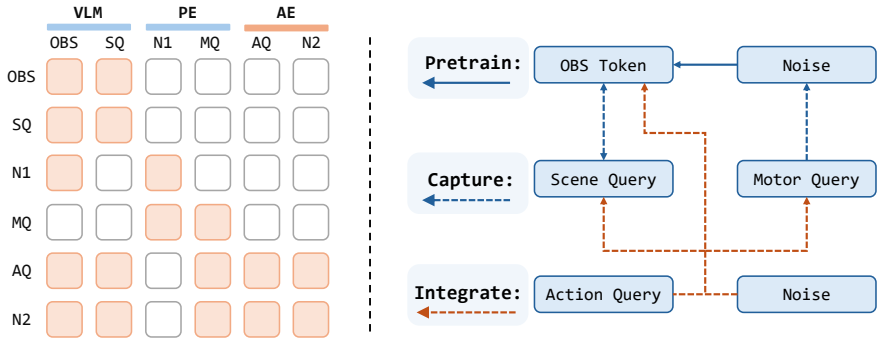


Figure 3: **Attention design of PriorVLA.** (**Left**) Attention mask over token groups in the VLM, Prior Expert (PE), and Adaptation Expert (AE). Orange cells indicate allowed attention and blank cells indicate blocked attention. OBS denotes original VLM input tokens, SQ Scene Queries, N1 PE noisy action tokens, MQ Motor Queries, AQ Action Queries, and N2 AE noisy action tokens. (**Right**) Information flow induced by the mask. The original VLM and PE attention paths are preserved; SQ and MQ capture scene and motor priors from pretrained forward paths, and AQ integrates them inside the AE for action generation.

The *Prior Expert* therefore serves as a read-only source of motor priors: its frozen weights preserve pretrained motor knowledge, while its internal denoising representations provide motor-prior features to the adapted policy, as described in Sec. 3.4.

3.4 Expert Queries

Structural separation preserves the pretrained AE, but does not by itself make its priors usable for downstream action generation. Since pretrained priors are distributed across layers and denoising steps, PriorVLA introduces *Expert Queries* as learnable token interfaces attached to different functional paths. They comprise three groups: *Scene Queries* capture task-relevant scene priors from the VLM, *Motor Queries* capture motor priors from the frozen *Prior Expert*'s denoising representations, and *Action Queries* integrate the captured priors inside the *Adaptation Expert* for action generation.

We implement these interfaces through masked attention over token groups. For any token group x at layer l, let $\mathbf{h}_x^l$ denote its hidden states, with $\mathbf{Q}_x^l = \mathbf{h}_x^l \mathbf{W}_Q^l$, $\mathbf{K}_x^l = \mathbf{h}_x^l \mathbf{W}_K^l$, and $\mathbf{V}_x^l = \mathbf{h}_x^l \mathbf{W}_V^l$. Fig. 3 summarizes the resulting attention mask and information flow; we then detail the three query groups and their roles in the adaptation path below.

Scene Queries. *Scene Queries* are learnable tokens inserted into the VLM input sequence alongside the original VLM input tokens. They participate in VLM self-attention and capture task-relevant scene priors from VLM representations:

$$\mathbf{h}_{sq}^{l+1} = \mathrm{Attn}\big(\mathbf{Q}_{sq}^l,\ \mathbf{K}_{obs}^l \| \mathbf{K}_{sq}^l,\ \mathbf{V}_{obs}^l \| \mathbf{V}_{sq}^l\big)\,. \tag{3}$$

Their layer-wise key-value caches provide a compact readout of visual, language, and state context, complementing the *Adaptation Expert*'s direct access to OBS.

Motor Queries. *Motor Queries* are learnable tokens appended to the frozen *Prior Expert* to capture motor priors from its denoising representations. To preserve the pretrained denoising path, the *Prior Expert*'s noisy action tokens do not attend to *Scene Queries* or *Motor Queries*:

$$\mathbf{h}_{\tilde{a}^{pe}}^{l+1} = \mathrm{Attn}\big(\mathbf{Q}_{\tilde{a}^{pe}}^l,\ \mathbf{K}_{obs}^l \| \mathbf{K}_{\tilde{a}^{pe}}^l,\ \mathbf{V}_{obs}^l \| \mathbf{V}_{\tilde{a}^{pe}}^l\big)\,. \tag{4}$$

Motor Queries self-attend and read only from these noisy action tokens:

$$\mathbf{h}_{mq}^{l+1} = \mathrm{Attn}\big(\mathbf{Q}_{mq}^l,\ \mathbf{K}_{mq}^l \| \mathbf{K}_{\tilde{a}^{pe}}^l,\ \mathbf{V}_{mq}^l \| \mathbf{V}_{\tilde{a}^{pe}}^l\big)\,. \tag{5}$$

This one-way design preserves the frozen denoising process while exposing motor priors through the layer-wise key-value caches of *Motor Queries*.

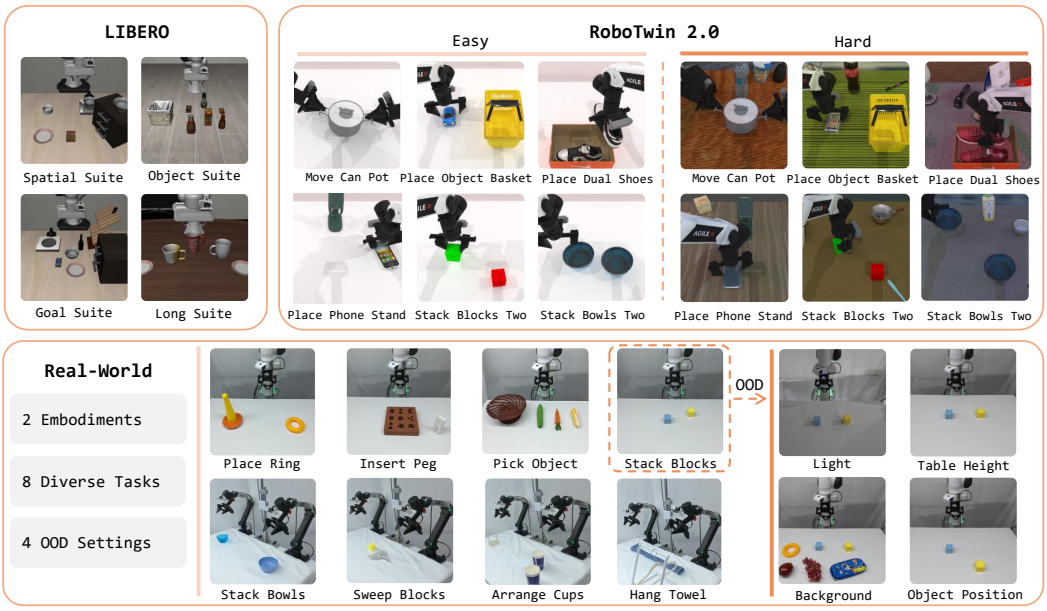


Figure 4: **Experimental overview.** We evaluate PriorVLA on RoboTwin 2.0 [22], LIBERO [25], and two real-world robot embodiments, covering ID/OOD generalization, data regimes, and component ablations.

Action Queries. *Action Queries* are learnable tokens inserted alongside the noisy action tokens of the *Adaptation Expert*. They integrate scene priors from *Scene Queries* and motor priors from *Motor Queries* inside the *Adaptation Expert* for downstream action generation:

$$\mathbf{h}^{l+1}_{aq,\tilde{a}^{ae}} = \mathrm{Attn}\big(\mathbf{Q}^{l}_{aq,\tilde{a}^{ae}},\ \mathbf{K}^{l}_{aq,\tilde{a}^{ae}} \| \mathbf{K}^{l}_{obs} \| \mathbf{K}^{l}_{sq} \| \mathbf{K}^{l}_{mq},\ \mathbf{V}^{l}_{aq,\tilde{a}^{ae}} \| \mathbf{V}^{l}_{obs} \| \mathbf{V}^{l}_{sq} \| \mathbf{V}^{l}_{mq}\big)\,. \tag{6}$$

The raw noisy-action key-value caches of the *Prior Expert* are excluded, so motor information is routed through the *Motor Query* interface rather than directly copied from frozen action states. Since *Action Queries* are not decoded as actions, they specialize in organizing multi-source priors inside the *Adaptation Expert*.

3.5 Training

Trainable parameters. We optimize the full *Adaptation Expert*, the three groups of *Expert Queries*, and the VLM vision encoder, while keeping all other VLM parameters and the *Prior Expert* frozen. Overall, PriorVLA updates approximately 25% of the parameters updated by full fine-tuning. This keeps pretrained prior sources in the frozen VLM components and *Prior Expert*, while allowing the vision encoder and adaptation branch to specialize to downstream data.

Objective. PriorVLA is trained with the standard flow-matching mean-squared error (MSE) objective, applied only to the denoising prediction of the *Adaptation Expert*. The *Prior Expert*'s prediction is discarded at both training and inference time and is never included in the loss. No auxiliary loss is imposed on the *Expert Queries* or the *Prior Expert*; all trainable components are optimized solely through the downstream action objective.

4 Experiments

To comprehensively evaluate PriorVLA, we conduct experiments on RoboTwin 2.0 [22], LIBERO [25], and eight real-world tasks across two robot embodiments. We organize the evaluation around five questions: (1) Can PriorVLA improve downstream adaptation over full fine-tuning and strong VLA baselines? (2) Does preserving pretrained priors improve OOD generalization? (3) Are the benefits stronger when downstream data coverage is limited? (4) Do these gains transfer to real-world robots across embodiments and evaluation conditions? (5) Are the gains explained by preserved pretrained priors and learnable query interfaces?

Table 1: **RoboTwin 2.0 simulation benchmark results.** Per-task success rates (%) on 13 tasks under standard training. Easy: in-distribution; Hard: out-of-distribution generalization. Best results are in **bold**.

Simulation Task	**DP**		**RDT**		π_0		$\pi_{0.5}$		**PriorVLA (Ours)**	
	Easy	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**
Grab Roller	**98**	0	74	43	96	80	97	**93**	**98**	**93**
Handover Mic	53	0	90	31	**98**	13	97	62	**98**	**84**
Lift Pot	39	0	72	9	84	36	67	25	**96**	**66**
Move Can Pot	39	0	25	12	58	21	**61**	36	**61**	**57**
Open Laptop	49	0	59	32	85	46	**91**	69	**91**	**83**
Pick Dual Bottles	24	0	42	13	57	12	55	17	**75**	**26**
Place Object Basket	15	0	33	17	16	2	62	38	**73**	**42**
Place Dual Shoes	8	0	4	4	15	0	40	18	**45**	**20**
Place Phone Stand	13	0	15	6	35	7	41	14	**65**	**35**
Put Bottles Dustbin	22	0	21	4	54	13	60	43	**64**	**45**
Put Object Cabinet	42	0	33	18	68	18	66	**53**	**73**	45
Stack Blocks Two	7	0	21	2	42	1	53	**24**	**70**	17
Stack Bowls Two	61	0	76	30	**91**	41	80	57	89	**73**
Average	36	0	44	17	62	22	67	42	**77** (+10)	**53** (+11)

Table 2: **Effect of training data scale on RoboTwin 2.0.** Average success rates (%) over 13 tasks across data regimes. Easy/Hard denote ID/OOD evaluation, and gains over $\pi_{0.5}$ are shown in parentheses.

Method	Easy			Hard		
	Few	Standard	Large	Few	Standard	Large
$\pi_{0.5}$ [5]	29%	67%	**89%**	20%	42%	59%
PriorVLA	**41%** (+12)	**77%** (+10)	88% (-1)	**31%** (+11)	**53%** (+11)	**65%** (+6)

4.1 Experimental Setup

Implementation details. PriorVLA is implemented on top of $\pi_{0.5}$ [5]. We train the Adaptation Expert, Expert Queries, and the VLM vision encoder, while keeping all other VLM parameters and the Prior Expert frozen. Unless otherwise stated, models are trained for 30k steps. Additional implementation details are provided in Appendix A and Appendix B.

Benchmarks and evaluation. We evaluate PriorVLA on RoboTwin 2.0 [22], LIBERO [25], and real-world tasks, as summarized in Fig. 4. RoboTwin 2.0 evaluates bimanual manipulation under Easy (ID) and Hard (OOD) modes; our standard setting uses 13 tasks with 50 clean demonstrations per task and 300 evaluation trials per setting. We further study data scaling on the same tasks under few-shot, standard, and large-data regimes. LIBERO evaluates four Franka manipulation suites: Spatial, Object, Goal, and Long, each with 10 tasks. Our real-world evaluation covers eight tasks across two embodiments, a single-arm Franka robot and a dual-arm AC-One platform, with both ID/OOD evaluation and few-shot training. OOD trials perturb four factors: Light, Background, Object Position, and Table Height. Detailed data and evaluation protocols are provided in Appendix B.1 and Appendix C.

4.2 Experimental Results

Simulated Evaluation on RoboTwin 2.0. As shown in Tab. 1, PriorVLA achieves **77%** Easy and **53%** Hard success under the standard setting, improving over $\pi_{0.5}$ [5] by +10 and +11 points, respectively. The gain is most pronounced under Hard OOD evaluation, where PriorVLA improves over $\pi_{0.5}$ [5] on most tasks and substantially outperforms prior methods such as Diffusion Policy [26], RDT [34], and π_0 [4]. As shown in Appendix D.6, PriorVLA outperforms FFT under all three training seeds on a six-task subset.

Table 3: **LIBERO benchmark results.** Average success rates (%) across four task suites. Best results in **bold**.

Methods	Spatial	Object	Goal	Long	Avg. Success
Diffusion Policy [26]	78.3	92.5	68.3	50.5	72.4
π_0-FAST [43]	96.4	96.8	88.6	60.2	85.5
DreamVLA [44]	97.5	94.0	89.5	89.5	92.6
GR00T-N1 [35]	94.4	97.6	93.0	90.6	93.9
π_0 [4]	96.8	98.8	95.8	85.2	94.1
UniVLA [11]	95.4	98.8	93.6	94.0	95.5
F1 [45]	98.2	97.8	95.4	91.3	95.7
DD-VLA [46]	97.2	98.6	97.4	92.0	96.3
GE-Act [47]	98.2	97.6	95.8	94.4	96.5
MemoryVLA [48]	98.4	98.4	96.4	93.4	96.7
$\pi_{0.5}$ [5]	98.8	98.2	98.0	92.4	96.9
OpenVLA-OFT [12]	97.6	98.4	97.9	94.5	97.1
PriorVLA (Ours)	**99.4**	**99.8**	**99.4**	**97.6**	**99.1**

Table 4: **LIBERO zero-shot transfer results.** Models are trained on LIBERO-10 and evaluated on Spatial, Object, and Goal. Average success rates (%) are reported. Coverage counts tasks with non-zero success.

Methods	Spatial	Object	Goal	Avg. Success	Coverage
FFT	5.8	**17.8**	2.8	8.80	6/30
LoRA [24]	10.4	4.0	11.8	8.73	7/30
RETAIN [19]	8.8	17.0	2.2	9.33	7/30
PriorVLA (Ours)	**21.0**	16.6	**19.6**	**19.07**	**12/30**

We further evaluate the same 13 tasks under few-shot, standard, and large-data regimes. As shown in Tab. 2, PriorVLA consistently improves Hard performance over $\pi_{0.5}$ [5] across all data scales, with gains of +11, +11, and +6 points. Easy performance improves under few-shot and standard data, while becoming comparable in the large-data regime; the OOD advantage remains across all regimes. This suggests that preserved priors are most beneficial when downstream coverage is limited, and remain useful for OOD generalization as data scales.

Simulated Evaluation on LIBERO. As shown in Tab. 3, PriorVLA achieves **99.1%** average success across four LIBERO suites, outperforming strong VLA baselines including OpenVLA-OFT [12] (97.1%) and $\pi_{0.5}$ [5] (96.9%). These results show that prior-preserving adaptation also improves standard in-distribution benchmark performance, even on a highly saturated benchmark.

Zero-shot task transfer. As shown in Tab. 4, PriorVLA achieves **19.07%** average success under zero-shot task transfer, outperforming FFT (8.80%), LoRA (8.73%), and RETAIN (9.33%). As described in Appendix D.5, all methods are trained on the 10 LIBERO-10 tasks and evaluated on the 30 tasks across LIBERO-Spatial, LIBERO-Object, and LIBERO-Goal without further parameter updates. PriorVLA also achieves non-zero success on **12 of 30** target tasks, compared with at most seven for these baselines. These results show that prior-preserving adaptation improves generalization beyond the downstream training tasks.

Real-World Robot Experiments. We evaluate whether PriorVLA transfers to real-world tasks across two embodiments. Standard-data training uses 100–300 demonstrations depending on task difficulty, while few-shot training uses only 10 demonstrations per task.

As shown in Tabs. 5 and 6, PriorVLA achieves the best real-world performance under both standard-data and few-shot settings. With standard data, it reaches **81%** ID and **57%** OOD success, improving over $\pi_{0.5}$ [5] by +12 and +16 points. With only 10 demonstrations per task, PriorVLA improves ID success from 24% to **48%** and OOD success from 10% to **32%**. These results show that PriorVLA transfers to real robots and provides larger gains when downstream demonstrations are scarce or evaluation conditions shift.

Table 5: **Real-robot standard-data results.** Success rates (%) on eight real-world tasks under ID and OOD evaluation. Average gains over $\pi_{0.5}$ are shown in parentheses.

Eval.	Method	Place Ring	Insert Peg	Pick Object	Stack Blocks	Stack Bowls	Sweep Blocks	Arrange Cups	Hang Towel	Avg. Success
ID	$\pi_{0.5}$ [5]	85	45	90	80	90	**100**	40	25	69
	GR00T-N1.7 [49]	60	20	75	40	**95**	95	30	10	53
	PriorVLA	**90**	**75**	**90**	**90**	**95**	95	**50**	**65**	**81** (+12)
OOD	$\pi_{0.5}$ [5]	45	10	55	50	70	75	15	10	41
	GR00T-N1.7 [49]	15	0	40	20	80	**85**	5	0	31
	PriorVLA	**55**	**30**	**60**	**65**	**85**	**85**	**25**	**50**	**57** (+16)

Table 6: **Real-robot few-shot results.** Success rates (%) when each task is trained with only 10 demonstrations. Average gains over $\pi_{0.5}$ are shown in parentheses.

Eval.	Method	Place Ring	Insert Peg	Pick Object	Stack Blocks	Stack Bowls	Sweep Blocks	Arrange Cups	Hang Towel	Avg. Success
ID	$\pi_{0.5}$ [5]	30	5	70	15	30	20	10	15	24
	PriorVLA	**75**	**30**	**85**	**65**	**55**	**35**	**15**	**25**	**48** (+24)
OOD	$\pi_{0.5}$ [5]	0	0	40	5	15	20	0	0	10
	PriorVLA	**40**	**20**	**50**	**55**	**30**	**30**	**10**	**20**	**32** (+22)

4.3 Ablation Study

We conduct component ablations on six RoboTwin 2.0 tasks and study Query configurations on four tasks, following the training and evaluation protocol in Sec. 4.2. Per-task component ablation results are provided in Appendix D.3.

Prior Expert. As shown in Tab. 7(a), PriorVLA achieves **49%** Hard success. Removing the Prior Expert or replacing it with a random frozen branch reduces success to 42% and 43%, respectively, showing the benefit of pretrained motor priors. Making the Prior Expert trainable also reduces Hard success to 44%, indicating that preserving a stable prior source is more effective than adapting both experts when generalizing beyond the downstream training distribution.

Expert Queries. As shown in Tab. 7(b), removing all Expert Queries reduces Hard success from **49%** to 28%. Among individual queries, removing Scene Queries causes the largest drop, to 30%, while removing Motor Queries or Action Queries reduces success to 42% and 43%, respectively. Appendix D.8 further analyzes how these Queries access information during action generation.

Query configuration. As shown in Tab. 8, halving the Query counts reduces Hard success from **61.5%** to 51.5%, which remains above FFT (42.5%). Query counts and learning rates are varied separately: the count ablation retains the 2/4/4 multipliers, while the learning rate ablation retains 64/16/64 SQ/MQ/AQ. Using the same learning rate as other trainable parameters for all Queries yields 59.3% Hard success. These results show that retaining the full Query counts improves OOD performance, while higher Query learning rates provide a further gain.

4.4 Adaptation Baselines

As shown in Tab. 9, PriorVLA achieves **48.3%** Hard success on six RoboTwin 2.0 tasks, outperforming FFT (33.7%), LoRA (26.0%), and RETAIN [19] (36.7%). As described in Appendix D.4, all methods share pretrained initialization, training data, and training budget. PriorVLA also maintains **77.3%** Easy success, close to LoRA (78.7%). These results show that PriorVLA maintains strong ID performance while achieving superior OOD generalization.

Table 7: **Component ablations.** Average success rates (%) are reported on a representative subset of six RoboTwin 2.0 tasks under Easy and Hard evaluation. Params denote trainable parameters; frozen parameters are not counted. The w/o PE row removes the PE-to-AE prior pathway; because PE information is routed only through MQ, it matches the w/o MQ setting in effect.

(a) Prior Expert ablation

Method	Variant	Params	Easy	Hard
PriorVLA	w/o PE	0.85B	75	42
	Random PE	0.85B	75	43
	Trainable PE	1.28B	73	44
	Full	0.85B	**77**	**49**

(b) Expert Queries ablation

Method	SQ	MQ	AQ	Easy	Hard
PriorVLA	×	×	×	61	28
	×	✓	✓	70	30
	✓	×	✓	75	42
	✓	✓	×	71	43
	✓	✓	✓	**77**	**49**

Table 8: **Query configuration ablations.** Average success rates (%) on four RoboTwin 2.0 tasks. LR Mult. denotes learning rate multipliers relative to the base learning rate, in SQ/MQ/AQ order.

Setting	SQ/MQ/AQ	LR Mult.	Easy	Hard
FFT	–	–	76.8	42.5
Half Query Counts	32/8/32	2/4/4	88.0	51.5
Uniform Query LR	64/16/64	1/1/1	87.5	59.3
PriorVLA (Ours)	64/16/64	2/4/4	**90.5**	**61.5**

Table 9: **Adaptation baselines.** Success rates (%) on six RoboTwin 2.0 tasks. Best results are shown in **bold**, and second-best results are underlined.

Method	Easy	Hard
FFT	66.2	33.7
LoRA [24]	**78.7**	26.0
Frozen VLM	35.3	13.7
RETAIN [19]	61.5	<u>36.7</u>
PriorVLA (Ours)	<u>77.3</u>	**48.3**

5 Conclusion and Limitations

In this work, we studied how to adapt pretrained Vision-Language-Action models without treating large-scale pretraining merely as initialization. We proposed PriorVLA, a prior-preserving adaptation framework that separates prior preservation from downstream specialization. Dual Action Experts retain a frozen Prior Expert as a read-only source of pretrained motor priors, while a trainable Adaptation Expert specializes to downstream action generation. Expert Queries further capture scene and motor priors from pretrained forward paths and integrate them inside the Adaptation Expert. Across RoboTwin 2.0, LIBERO, and real-world tasks, PriorVLA improves downstream adaptation over full fine-tuning and strong VLA baselines, with the largest gains under OOD and few-shot settings. Ablations further show that these gains depend on both preserved pretrained priors and learnable query interfaces. These results highlight prior preservation as a promising principle for data-efficient and generalizable VLA adaptation.

Limitations. While PriorVLA demonstrates strong adaptation across simulation and real-world settings, several aspects remain to be further studied. Our RoboTwin 2.0 results are reported on a representative 13-task subset rather than the full 50-task benchmark because each task requires a separate model. In real-world evaluation, OOD factors are applied jointly, which reflects deployment-like shifts but does not disentangle the effect of each factor. As reported in Appendix D.7, PriorVLA introduces additional inference computation because the frozen Prior Expert is executed during denoising; although this overhead is manageable in our chunked-control setting, more efficient implementations or cached prior readouts could further reduce deployment cost. Finally, how scene and motor priors evolve across layers and denoising steps remains to be further studied.

Acknowledgments and Disclosure of Funding

This work was partly supported by the National Natural Science Foundation of China under Grant 62403012, Grant 62673073, and Zhongguancun Academy under Grant C20250503.

References

[1] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[2] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 2023.

[3] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2025.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025.

[5] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 2025.

[6] 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, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1723–1736. PMLR, 2023.

[7] 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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[8] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R. Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[9] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[10] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[11] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems*, 2025.

[12] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025.

[13] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[14] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action model. In *Proceedings of Robotics: Science and Systems*, 2025.

[15] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

[16] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

[17] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*, 2025.

[18] Chengyue Huang, Mellon M. Zhang, Robert Azarcon, Glen Chou, and Zsolt Kira. MAPS: Preserving vision-language representations via module-wise proximity scheduling for better vision-language-action generalization. *arXiv preprint arXiv:2511.19878*, 2025.

[19] Yajat Yadav, Zhiyuan Zhou, Andrew Wagenmaker, Karl Pertsch, and Sergey Levine. Robust fine-tuning of vision-language-action robot policies via parameter merging. In *International Conference on Learning Representations*, 2026.

[20] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1820–1864. PMLR, 2023.

[21] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of household tasks for generalist robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[22] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

[23] Haoran Geng, Feishi Wang, Songlin Wei, Yuyang Li, Bangjun Wang, Boshi An, Haozhe Lou, Charlie Tianyue Cheng, Peihao Li, Haozhe Chen, Yutong Liang, Yuxi Qian, Jiageng Mao, Weikang Wan, Yiran Geng, Mingtong Zhang, Jiangran Lyu, Siheng Zhao, Jiazhao Zhang, Chaoyi Xu, Jialiang Zhang, Chengyang Zhao, Haoran Lu, Yufei Ding, Ran Gong, Yuran Wang, Yuxuan Kuang, Ruihai Wu, Baoxiong Jia, Hao Dong, Siyuan Huang, Yue Wang, Jitendra Malik, and Pieter Abbeel. RoboVerse: A unified platform, benchmark and dataset for scalable and generalizable robot learning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025.

[24] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[25] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[26] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[27] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[28] Henry Zheng, Hao Shi, Qihang Peng, Yong Xien Chng, Rui Huang, Yepeng Weng, Zhongchao Shi, and Gao Huang. Densegrounding: Improving dense language-vision semantics for egocentric 3d visual grounding. *arXiv preprint arXiv:2505.04965*, 2025.

[29] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 785–799. PMLR, 2023.

[30] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3d object manipulation. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 694–710. PMLR, 2023.

[31] Hao Shi, Bin Xie, Yingfei Liu, Yang Yue, Tiancai Wang, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Spatialactor: Exploring disentangled spatial representations for robust robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 8969–8977, 2026.

[32] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, Shu Jiang, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[33] Huailiang Ma, Aiguo Song, Mutian He, Mingyu Li, Yibing Yan, and Linhu Wei. Autotrialgen: Automated data generation from few human demonstrations via trajectory annotation and simulation trials. *IEEE Robotics and Automation Letters*, 11(6):6935–6942, 2026.

[34] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, 2025.

[35] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loïc Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[36] Gemini Robotics Team. Gemini Robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[37] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[38] Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations*, 2026.

[39] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[40] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[41] Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. RICL: Adding in-context adaptability to pre-trained vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 5022–5038. PMLR, 2025.

[42] Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. ICRT: In-context imitation learning via next-token prediction. In *IEEE International Conference on Robotics and Automation*, 2025.

[43] 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.

[44] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. *arXiv preprint arXiv:2507.04447*, 2025.

[45] Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025.

[46] Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Tian Nian, Liuao Pei, Shunbo Zhou, Xiaokang Yang, Jiangmiao Pang, Yao Mu, and Ping Luo. Discrete Diffusion VLA: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072*, 2025.

[47] Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng Yan, Maoqing Yao, and Guanghui Ren. Genie Envisioner: A unified world foundation platform for robotic manipulation. In *International Conference on Learning Representations*, 2026.

[48] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026.

[49] NVIDIA. NVIDIA Isaac GR00T N1.7: Open foundation model for generalized humanoid robot reasoning and skills. `https://huggingface.co/nvidia/GR00T-N1.7-3B`, 2026. Model card.

A Implementation Notes for PriorVLA

The main paper presents the architecture of PriorVLA, including Dual Action Experts, Expert Queries, the attention design, and the training objective in Sec. 3. This section provides implementation-oriented details that are useful for reproducibility but omitted from the main text for clarity.

A.1 Base VLA Backbone

PriorVLA is built on a pretrained $\pi_{0.5}$-style VLA backbone [5], which couples a VLM with a flow-matching action expert. The VLM consists of a SigLIP-style vision encoder and a Gemma-2B language backbone, and encodes multi-view images, language prompts, and proprioceptive states. The action expert follows a Gemma-300M-style transformer architecture and denoises future action chunks for continuous action generation. PriorVLA keeps this pretrained backbone structure while adding Dual Action Experts and Expert Queries for prior-preserving adaptation.

A.2 Dual Action Experts

The Prior Expert and Adaptation Expert are both initialized from the pretrained AE. The Prior Expert is frozen and serves as a read-only source of motor-prior representations, while the Adaptation Expert is trainable and drives the denoising trajectory that produces the final action chunk.

At each denoising step, both experts receive the same current noisy action chunk. The Adaptation Expert updates the action trajectory, and the updated chunk is used as the input to both experts at the next step. The Prior Expert is executed along this trajectory, but its denoising output is discarded and is not used in the loss. Its internal representations are instead exposed through Motor Queries as motor-prior features for the Adaptation Expert.

A.3 Expert Queries

PriorVLA uses three groups of learnable Expert Queries. Scene Queries are provided as additional input tokens to the VLM, alongside multi-view image tokens, prompt tokens, and proprioceptive-state tokens. They form a compact interface for capturing scene-prior features.

Motor Queries are provided as additional input tokens to the frozen Prior Expert at each denoising step. They are not decoded as actions and do not update the noisy action trajectory; instead, they read motor-prior features from the Prior Expert forward path.

Action Queries are provided as additional input tokens to the trainable Adaptation Expert at each denoising step. They are not decoded as actions; instead, they integrate scene-prior features from Scene Queries and motor-prior features from Motor Queries to guide the Adaptation Expert's denoising prediction.

A.4 Attention-Mask Implementation

The attention mask is implemented with the block structure shown in Fig. 3. Tokens within the same block use bidirectional self-attention, while later blocks can attend to earlier blocks but not vice versa. This allows PriorVLA to preserve the pretrained action path of the frozen Prior Expert while adding one-way query interfaces for prior capture. The original VLM input tokens (OBS) and Scene Queries (SQ) belong to the same bidirectional attention block, so each can attend to the other.

For the frozen Prior Expert, noisy action tokens follow the original pretrained action path and are not allowed to attend to Motor Queries. Motor Queries are placed in a later readout block, so they can attend to the Prior Expert's noisy action tokens, but they are blocked from directly attending to the VLM prefix or Scene Queries. This makes Motor Queries focus on action-side representations from the frozen Prior Expert. In practice, this is important because the VLM prefix contains many more tokens than the Prior Expert action path; allowing Motor Queries to attend to the VLM prefix can make them dominated by scene features rather than motor-prior features.

For the trainable Adaptation Expert, Action Queries and noisy action tokens are placed in the same adaptation block, allowing bidirectional interaction within the Adaptation Expert. The Adaptation Expert can attend to the original VLM features, Scene Query features, and Motor Query features, but

raw noisy-action states from the Prior Expert are excluded. Empirically, direct access to these raw Prior Expert action states makes training less stable, while routing motor-prior information through Motor Queries provides a compact and more stable interface.

B Training Details

B.1 Training Data

RoboTwin 2.0. For RoboTwin 2.0 [22], we follow the official setting and train one model per task. Given the cost of training the full 50-task benchmark, we use the 13 tasks reported in the official main table, rather than selecting tasks based on PriorVLA performance. The standard setting uses the official 50 clean demonstrations provided for each task. For the few-shot and large-data settings, we use the RoboTwin 2.0 data-generation pipeline to construct 10 and 250 clean demonstrations per task, respectively.

LIBERO. For LIBERO [25], we train one model for each suite: Spatial, Object, Goal, and Long. Each suite contains 10 tasks, and LIBERO provides 50 demonstrations per task for training.

Real-world tasks. We collect real-world demonstrations for eight tasks across two robot embodiments: a single-arm Franka robot and a dual-arm AC-One platform (Fig. 5). The Franka setup uses one third-person Intel RealSense D435 camera and one wrist Intel RealSense D435 camera. The AC-One setup uses one top-view Intel RealSense D435 camera and two Intel RealSense D405 wrist cameras, one for each arm. This setup covers both single-arm and dual-arm data collection with different camera configurations. In the standard-data setting, simple, medium, and hard tasks use 100, 200, and 300 demonstrations per task, respectively. The few-shot setting uses 10 demonstrations per task for all real-world tasks.

Table 10: **Real-world task suite used for training.** Demonstrations refer to the standard-data setting; the few-shot setting uses 10 demonstrations per task.

Level	Task	Robot	Demonstrations
Simple	Place Ring	Franka	100
Simple	Stack Blocks	Franka	100
Simple	Stack Bowls	AC-One	100
Medium	Pick Object	Franka	200
Medium	Insert Peg	Franka	200
Medium	Sweep Blocks	AC-One	200
Medium	Arrange Cups	AC-One	200
Hard	Hang Towel	AC-One	300

B.2 Hyperparameters

Across all experiments, we use AdamW with global-norm gradient clipping at 1.0. Trainable parameters are optimized in float32, and frozen pretrained parameters are stored in bfloat16. Unless otherwise specified, all runs use random seed 42. PriorVLA applies grouped learning-rate multipliers to four parameter groups: Scene Queries, Motor Queries, Action Queries, and the remaining trainable parameters. The corresponding multipliers are 2.0, 4.0, 4.0, and 1.0. We use 64 Scene Queries, 16 Motor Queries, and 64 Action Queries for RoboTwin 2.0 and real-world experiments, and 8, 2, and 8, respectively, for LIBERO. The benchmark-specific hyperparameters are summarized in Table 11.

RoboTwin 2.0. For RoboTwin 2.0 [22], we train a separate model for each task for 30k steps with action horizon $H = 50$. The few-shot and standard-data settings use global batch size 32, base peak learning rate 2.5×10^{-5}, and base decay learning rate 2.5×10^{-6}. The large-data setting uses global batch size 128 and doubles the base peak and decay learning rates to 5.0×10^{-5} and 5.0×10^{-6}. All RoboTwin 2.0 runs use 1k warmup steps, cosine decay over 30k steps, and EMA decay 0.99.

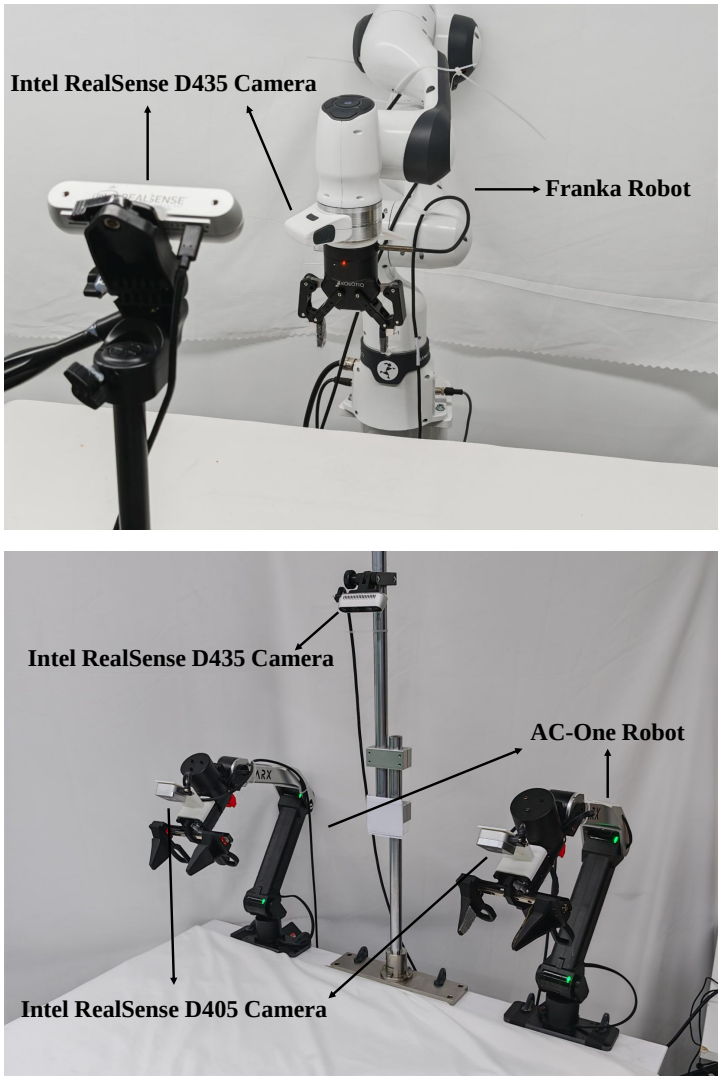


Figure 5: **Real-world platforms used in our experiments.** Top: the single-arm Franka setup with one third-person and one wrist Intel RealSense D435 camera. Bottom: the dual-arm AC-One setup with one top-view Intel RealSense D435 camera and two Intel RealSense D405 wrist cameras.

LIBERO. For LIBERO [25], we train a separate model for each suite for 30k steps with action horizon $H = 10$ and global batch size 256. To align with the standard $\pi_{0.5}$ [5] LIBERO [25] setting, we keep the official training schedule unchanged: 10k warmup steps, peak and decay learning rates of 5.0×10^{-5}, 1M decay steps, and EMA decay 0.999.

Real-world tasks. For real-world tasks, we train one model per task for 30k steps with action horizon $H = 50$ and global batch size 32. We use the same base learning-rate schedule as RoboTwin 2.0: 1k warmup to 2.5×10^{-5}, followed by cosine decay to 2.5×10^{-6} over 30k steps. The EMA decay is 0.99.

B.3 Training Procedure

Each training sample is constructed from a demonstration trajectory and contains multi-view RGB observations, a language prompt, proprioceptive states, and a future action chunk. Image frames are aligned with state-action records, and future action chunks are constructed according to the benchmark-specific action horizon H. Actions are represented as delta actions and normalized using

Table 11: **Benchmark-specific training hyperparameters.** RoboTwin 2.0 values correspond to the few-shot and standard-data settings; the large-data setting uses global batch size 128 and doubled base learning rates.

Hyperparameter	RoboTwin 2.0	LIBERO	Real-world
Training granularity	Per task	Per suite	Per task
Training steps	30k	30k	30k
Global batch size	32	256	32
Action horizon H	50	10	50
Warmup steps	1k	10k	1k
Base peak LR	2.5×10^{-5}	5.0×10^{-5}	2.5×10^{-5}
Base decay LR	2.5×10^{-6}	5.0×10^{-5}	2.5×10^{-6}
Decay steps	30k	1M	30k
EMA decay	0.99	0.999	0.99

training-set statistics. For the dual-arm AC-One platform, we concatenate the left- and right-arm delta actions into a joint 14-D action representation.

Images are resized to 224×224 before being fed into the policy. During training, we use the default $\pi_{0.5}$ image preprocessing and augmentation pipeline, and disable image augmentations during evaluation.

For RoboTwin 2.0 and real-world experiments, we report the final checkpoint after 30k training steps. LIBERO follows the standard $\pi_{0.5}$ LIBERO checkpointing and evaluation recipe.

Compute resources. Main experiments are trained on 8 NVIDIA GPUs. RoboTwin 2.0 experiments are trained on H20 GPUs, while LIBERO and real-world experiments are trained on A100 GPUs. Table 12 reports representative wall-clock time per trained model and the memory capacity of the GPUs used in our experiments.

Table 12: **Representative compute resources.** Wall-clock time is reported per trained model and is approximate. Mem./GPU denotes the memory capacity of each GPU. The $\pi_{0.5}$ LIBERO baseline is taken from the official report and was not re-trained in our compute accounting.

Setting	Method	GPUs	Mem./GPU	Batch	Steps	Time
RoboTwin 2.0 standard	$\pi_{0.5}$	8 H20	96GB	32	30k	∼6.8h
	PriorVLA	8 H20	96GB	32	30k	∼5.6h
LIBERO	$\pi_{0.5}$	–	–	–	–	–
	PriorVLA	8 A100	80GB	256	30k	∼23.6h
Real-world tasks	$\pi_{0.5}$	8 A100	80GB	32	30k	∼6.5h
	PriorVLA	8 A100	80GB	32	30k	∼5.0h

C Evaluation Protocol

This section describes the evaluation protocols used for RoboTwin 2.0, LIBERO, and real-world experiments. We report success rate as the primary metric. RoboTwin 2.0 and LIBERO use their official success signals, while real-world experiments use task-specific success criteria.

C.1 RoboTwin 2.0 Evaluation

Protocol. For RoboTwin 2.0, we evaluate the same 13 tasks used for training under the official Easy and Hard modes. Easy serves as the ID setting, and Hard serves as the OOD setting with randomized scene factors. For each method, we evaluate the final 30k checkpoint with 300 rollouts per task for each mode. Success is measured by the official RoboTwin 2.0 simulator signal.

Easy and Hard modes. Easy and Hard modes use the same robot embodiment, camera setup, and observation/state inputs, and differ only in domain randomization. The Easy mode disables scene

randomization and serves as the ID setting. The Hard mode enables background randomization, table clutter, lighting variation, and table-height perturbation, and serves as the OOD setting. The main differences are summarized in Table 13.

Table 13: **RoboTwin 2.0 Easy and Hard modes.** The Easy mode is the clean ID setting, while the Hard mode enables domain randomization and is used as the OOD setting.

Field	Easy mode / ID	Hard mode / OOD
Domain randomization		
Background randomization	Off	On
Table clutter	Off	On
Clean background rate	1.00	0.02
Table-height perturbation	0	0.03
Lighting randomization	Off	On
Extreme lighting rate	0	0.02
Shared setup		
Robot embodiment	Aloha-AgileX	Aloha-AgileX
Head / wrist camera	D435 / D435	D435 / D435
RGB observation	Yes	Yes
End-effector state / qpos	Yes / Yes	Yes / Yes

C.2 LIBERO Evaluation

For LIBERO, we evaluate the four suites used in the main paper: Spatial, Object, Goal, and Long. Each suite contains 10 tasks, and each task is evaluated with 50 rollouts, yielding 500 rollouts per suite. We use the initial states provided by LIBERO for each task, and success is measured by the LIBERO environment success signal. The evaluation seed is 7, and the policy replans every 10 environment steps.

For checkpointing, we follow the standard $\pi_{0.5}$ LIBERO evaluation recipe. Checkpoints from 20k steps onward are evaluated at fixed 1k-step intervals, and the best checkpoint is reported.

C.3 Real-World Evaluation

Protocol. For real-world evaluation, we evaluate the final 30k checkpoint of each trained policy on 20 ID trials and 20 OOD trials per task. The same policy is evaluated under both settings without retraining. For all real-world tasks, the policy predicts an action chunk of length $H = 50$, executes the first 15 actions, and then re-queries the policy with a new observation. Success rates are reported separately for ID and OOD trials.

ID and OOD settings. ID trials follow the nominal task setup used during data collection. OOD trials use a harder test-time setup that jointly changes four dimensions: Light, Background, Object Position, and Table Height. All four changes are applied together in each OOD trial, while the task instruction and robot platform remain unchanged. All compared methods are evaluated under the same ID/OOD protocol. The OOD dimensions are summarized in Table 14.

Table 14: **Real-world OOD dimensions.** Each OOD trial jointly applies all four dimensions while keeping the task instruction and robot platform unchanged.

Dimension	OOD Perturbation
Light	The scene is evaluated under darker lighting conditions.
Background	Distracting objects are added to the background or workspace.
Object Position	Task objects are initialized at positions different from the nominal setup.
Table Height	The table is raised by 2 cm relative to the nominal setup.

Success criteria. A real-world trial is counted as successful only if the task-specific completion condition is satisfied before the trial ends. A trial is counted as a failure if the task is not completed, the object is dropped irrecoverably, a safety stop or collision occurs, or human intervention is required. The task-specific success criteria are listed in Table 15.

Table 15: **Task-specific success criteria for real-world evaluation.**

Task	Success Criterion
Place Ring	The ring is inserted onto the vertical peg.
Insert Peg	The peg is inserted into the matching slot.
Pick Object	The target object is placed into the basket after release.
Stack Blocks	The target block is stacked on the other block and remains stable after release.
Stack Bowls	The bowls are stacked together, and the upper bowl does not touch the table.
Sweep Blocks	The blocks are pushed into the container and do not remain on the table.
Arrange Cups	The cups are stacked together and fully placed into the rack.
Hang Towel	The towel remains on the hanger after the hanger is lifted and returned.

Table 16: Full per-task RoboTwin 2.0 results across data regimes. Success rates (%) are reported under Easy and Hard evaluation. Best values within each data regime are shown in bold, and PriorVLA columns are shaded.

Task	$\pi_{0.5}$ Few		PriorVLA Few		$\pi_{0.5}$ Standard		PriorVLA Standard		$\pi_{0.5}$ Large		PriorVLA Large	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
Grab Roller	41	46	**79**	**73**	97	**93**	**98**	**93**	**100**	93	**100**	**98**
Handover Mic	85	60	**88**	**77**	97	62	**98**	**84**	**99**	65	**99**	**83**
Lift Pot	8	5	**51**	**28**	67	25	**96**	**66**	99	51	**100**	**69**
Move Can Pot	45	**32**	**47**	28	**61**	36	**61**	**57**	90	**70**	**92**	68
Open Laptop	69	37	**78**	**60**	**91**	69	**91**	**83**	**98**	75	96	**95**
Pick Dual Bottles	3	2	**10**	**19**	55	17	**75**	**26**	88	42	**93**	**45**
Place Object Basket	31	15	**34**	**20**	62	38	**73**	**42**	**76**	**42**	72	40
Place Dual Shoes	1	1	**4**	**2**	40	18	**45**	**20**	**79**	**42**	68	38
Place Phone Stand	1	4	**8**	**6**	41	14	**65**	**35**	**74**	36	**74**	**53**
Put Bottles Dustbin	16	8	**21**	**18**	60	43	**64**	**45**	**83**	65	79	**66**
Put Object Cabinet	25	17	**32**	**21**	66	**53**	**73**	45	**91**	67	86	**70**
Stack Blocks Two	8	4	**13**	**5**	53	**24**	**70**	17	87	34	**91**	**43**
Stack Bowls Two	43	27	**67**	**43**	80	57	**89**	**73**	**94**	82	89	**84**
Average	29	20	**41**	**31**	67	42	**77**	**53**	**89**	59	88	**65**

D Additional Results

This section reports additional experimental records that are not fully expanded in the main paper. We provide detailed RoboTwin 2.0 results and additional analysis of LIBERO zero-shot task transfer. The full per-task RoboTwin 2.0 results across few-shot, standard, and large-data settings are reported in Table 16. The full per-task ablation results on the six-task RoboTwin 2.0 subset are reported in Table 18.

D.1 RoboTwin 2.0 Results across Data Regimes

The main paper reports the average RoboTwin 2.0 data-regime comparison. Here, we provide the per-task results across few-shot, standard, and large-data settings in Table 16. These results show that PriorVLA improves OOD performance across data regimes. In the large-data setting, the in-distribution Easy average of PriorVLA is close to $\pi_{0.5}$, while Hard performance remains better, suggesting that additional demonstrations reduce the need for preserved priors on ID scenes but do not remove the need for preserved priors under distribution shift.

D.2 Consistency and Sign-Test Analysis

Beyond aggregate averages, we analyze whether PriorVLA's gains are consistent across task-setting pairs. For RoboTwin 2.0, each pair corresponds to one task under a specific data regime and evaluation mode. For real-world experiments, each pair corresponds to one task under either the ID or OOD evaluation setting. As shown in Table 17, PriorVLA improves on a large majority of non-tie task-setting pairs. The gains are therefore not driven by a single task or a single setting, but are consistent across RoboTwin 2.0 few-shot, standard, and OOD settings, as well as standard-data and few-shot real-world evaluations. For real-world evaluations, each ID or OOD aggregate contains 8 tasks with 20 trials per task, yielding 160 trials per evaluation condition.

Table 17: **Consistency of improvements over $\pi_{0.5}$.** We count task-setting pairs where PriorVLA improves over $\pi_{0.5}$. Ties are excluded when computing the sign-test statistics. Avg. Gain denotes the aggregate average success-rate difference in points. Sign-test p denotes the one-sided exact binomial p-value under the null hypothesis that either method is equally likely to perform better.

Setting	**Non-tie Pairs**	**Improved**	**Avg. Gain (pts)**	**Sign-test p**
RoboTwin 2.0 few-shot Easy	13	13	+12	1.2×10^{-4}
RoboTwin 2.0 few-shot Hard	13	12	+11	1.7×10^{-3}
RoboTwin 2.0 standard Easy	11	11	+10	4.9×10^{-4}
RoboTwin 2.0 standard Hard	12	10	+11	1.9×10^{-2}
RoboTwin 2.0 large-data Hard	13	10	+6	4.6×10^{-2}
Real-world standard ID/OOD	15	14	+14	4.9×10^{-4}
Real-world few-shot ID/OOD	16	16	+23	1.5×10^{-5}

D.3 RoboTwin 2.0 Ablation Results

We conduct ablations on a representative subset of six RoboTwin 2.0 tasks: Handover Mic, Lift Pot, Place Dual Shoes, Place Phone Stand, Stack Blocks Two, and Stack Bowls Two. These tasks cover grasping, placement, stacking, and longer-horizon manipulation. The main paper reports the average ablation comparison. Here, we provide the per-task Easy and Hard results in Table 18, including Prior Expert controls, Expert Query ablations, and the vision-encoder ablation.

Table 18: Per-task ablation results on the six-task RoboTwin 2.0 subset. Success rates (%) are reported under Easy and Hard evaluation. Best values in each column are shown in bold, and the PriorVLA row is shaded.

Variant	Average		Handover Mic		Lift Pot		Place Dual Shoes		Place Phone Stand		Stack Blocks Two		Stack Bowls Two	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
Baseline	63	33	97	62	67	25	40	18	41	14	53	**24**	80	57
Random PE	75	43	96	79	95	48	43	11	61	34	69	15	87	72
Trainable PE	73	44	**98**	76	94	58	34	10	59	34	66	14	88	74
w/o SQ/MQ/AQ	61	28	94	27	93	48	25	12	45	8	30	15	80	58
w/o AQ	71	43	95	82	92	47	30	11	55	30	67	14	88	75
w/o SQ	70	30	**98**	26	95	49	32	3	52	20	59	14	83	70
w/o MQ	75	42	**98**	80	**96**	37	40	14	60	34	**70**	15	86	71
Frozen ViT	65	37	87	66	83	20	24	10	45	**35**	65	14	87	**76**
PriorVLA	**77**	**49**	**98**	**84**	**96**	**66**	**45**	**20**	**65**	**35**	**70**	17	**89**	73

Analysis. The Prior Expert controls support the role of a pretrained and preserved prior source. The w/o PE and w/o MQ variants have the same effect on action generation because PE information is routed to the Adaptation Expert only through Motor Queries. Thus, w/o PE removes the prior source, while w/o MQ keeps the frozen branch but disables its usable interface. Their similar performance confirms that the Prior Expert must be connected through a learnable Motor Query interface to benefit adaptation. Random PE reaches 75% Easy and 43% Hard, close to the w/o MQ variant, indicating that a random frozen branch does not provide useful motor priors. Trainable PE reaches 73% Easy and 44% Hard, also below full PriorVLA, suggesting that updating the prior source weakens its usefulness. Removing all Expert Queries reduces Hard performance from 49% to 28%, showing that keeping a frozen Prior Expert alone is insufficient without learnable interfaces. Among single-query ablations, removing Scene Queries causes the largest Hard degradation, while removing Motor Queries or Action Queries also reduces OOD performance. Freezing the vision encoder decreases both Easy and Hard performance, showing that visual adaptation remains complementary to prior preservation.

The Query configuration ablations, initialization comparison, and attention analysis use the same four RoboTwin 2.0 tasks: Handover Mic, Lift Pot, Stack Blocks Two, and Stack Bowls Two. These experiments use separate runs from the six-task component ablations.

On this subset, initializing the VLM from PaliGemma achieves 60.8% Easy and 2.8% Hard success, compared with 90.5% and 61.5% when initializing it from $\pi_{0.5}$ as in PriorVLA. Both settings use the same architecture, PE and AE initialization, Query counts, data, and training budget.

Table 19: Adaptation baselines on the six-task RoboTwin 2.0 subset. Average success rates (%) are reported under Easy and Hard evaluation. Best values in each column are shown in bold, and the PriorVLA row is shaded.

Method	Easy	Hard
FFT	66.2	33.7
LoRA [24]	**78.7**	26.0
Frozen VLM	35.3	13.7
First-4 VLM Tuning	70.8	32.4
RETAIN [19] ($\alpha = 0.95$)	54.5	32.2
RETAIN [19] ($\alpha = 0.98$)	61.5	36.7
PriorVLA (Ours)	77.3	**48.3**

Table 20: RoboTwin 2.0 results across three training seeds. Mean success rates (%) and sample standard deviations are reported on the six-task subset. Best results are shown in bold, and the PriorVLA row is shaded.

Method	Easy	Hard
FFT	64.6 ± 1.4	33.7 ± 0.4
PriorVLA (Ours)	$\mathbf{77.3 \pm 0.1}$	$\mathbf{49.0 \pm 0.6}$

D.4 Adaptation Baselines

We compare adaptation baselines on the six-task RoboTwin 2.0 subset described in Appendix D.3, using the same pretrained initialization, training data, and training budget. Table 19 provides the full comparison corresponding to Table 9, including First-4 VLM Tuning and both RETAIN configurations. This comparison uses separate runs from the component ablations in Table 18.

LoRA follows the default openpi configuration, with ranks of 16 and 32 for the VLM language backbone and action expert, respectively, applied to attention and feed-forward modules. Frozen VLM freezes the entire VLM, including the vision encoder, and trains only the action expert. First-4 VLM Tuning updates the first four VLM transformer layers while freezing the remaining VLM transformer layers. RETAIN interpolates pretrained and finetuned weights as $\theta = (1 - \alpha)\theta_{\text{pre}} + \alpha\theta_{\text{ft}}$.

D.5 Zero-shot Task Transfer

For the zero-shot transfer experiment in Table 4, all methods are adapted on LIBERO-10 and evaluated on LIBERO-Spatial, LIBERO-Object, and LIBERO-Goal, without target-suite demonstrations or further parameter updates. RETAIN [19] uses $\alpha = 0.9$ in this comparison.

Removing Scene Queries reduces average success from 19.07% to 16.93%, supporting their contribution to generalization beyond the downstream training tasks.

D.6 Results across Training Seeds

We train FFT and PriorVLA with three independent random seeds on the six-task RoboTwin 2.0 subset described in Appendix D.3, following the same evaluation protocol. Table 20 reports mean success rates and sample standard deviations. PriorVLA outperforms FFT under every training seed on both Easy and Hard.

D.7 Inference Cost

We measure model-only inference on an NVIDIA A100 using BF16, batch size 1, and 10 denoising steps. Latency is averaged over 300 action chunks after warm-up, with 50 actions per chunk.

As shown in Table 21, PriorVLA increases inference latency by 40.5% and peak GPU memory by 0.88 GiB. The frozen Prior Expert is executed alongside the Adaptation Expert at each denoising step, while vision encoder and VLM computation are shared.

Table 21: Model-only inference cost. Each action chunk contains 50 actions.

Method	Latency (ms/chunk)	Throughput (chunks/s)	Peak VRAM (GiB)
$\pi_{0.5}$	88.59	11.29	6.58
PriorVLA	124.51	8.03	7.46

Table 22: Attention allocation of Expert Queries and AE action tokens. Values are percentages of attention assigned to each source. The final row reports the combined attention of AE action tokens to SQ, MQ, and AQ.

Tokens	Source	Attention (%)
SQ	Image OBS	25.4
	Text/State OBS	40.9
	SQ self	33.7
MQ	Frozen PE states	55.7
	MQ self	44.3
AQ	Direct OBS	38.2
	SQ	5.6
	MQ	4.3
	AQ self + AE action tokens	51.8
AE action tokens	SQ + MQ + AQ	12.8

D.8 Attention Analysis

We analyze attention after masking and softmax over all action chunks from eight Easy/Hard episodes across the four-task subset described in Appendix D.3. We average attention over heads, Query tokens of the same type, and the final four layers.

As shown in Table 22, SQ attends to both image and text/state context, while MQ reads frozen PE states. AQ assigns most attention to direct OBS and action states, with SQ and MQ providing an additional input path.

E Task Details

To ensure comprehensive evaluation across simulation and real-world settings, we summarize the tasks and language instructions used in our experiments. The evaluation covers real-world manipulation on Franka and AC-One, single-arm simulation in LIBERO, and bimanual simulation in RoboTwin 2.0.

E.1 Real-World Tasks

The real-world benchmark consists of eight manipulation tasks across a single-arm Franka robot and a dual-arm AC-One platform. These tasks cover precise insertion, object picking, stacking, sweeping, rearrangement, and deformable-object manipulation. Each task is evaluated under both ID and OOD conditions. ID trials use the nominal task setup, while OOD trials perturb lighting, background objects, initial object locations, and table height. Table 23 summarizes the platform assignment and representative instruction for each real-world task.

E.2 LIBERO Tasks

The LIBERO benchmark evaluates single-arm manipulation in simulation with a Franka robot. We use four suites: Spatial, Object, Goal, and Long. LIBERO-Spatial consists of tasks where the same object is placed across varying target positions, stressing spatial grounding and positional generalization. LIBERO-Object focuses on handling diverse objects within similar scene layouts, testing object-centric recognition and manipulation. LIBERO-Goal contains heterogeneous operations such as opening containers, placing objects into or onto target regions, and interacting with appliances, performed in a shared environment. LIBERO-Long introduces extended tasks that require multiple sub-goals across different scenes, providing a more challenging setting for long-horizon execution and

Table 23: Real-world task details. We list the task prompts used for real-world evaluation.

Task	Platform	Prompt
Place Ring	Franka	Pick up the ring and place it onto the vertical peg.
Insert Peg	Franka	Pick up the object and insert it into the matching slot.
Pick Object	Franka	Pick up the target object and place it into the basket.
Stack Blocks	Franka	Pick up one block and place it on top of the other block.
Stack Bowls	AC-One	Pick up the two bowls and stack one bowl on top of the other.
Sweep Blocks	AC-One	Use one arm to hold the container and the other arm to push all objects into it.
Arrange Cups	AC-One	Use the left arm to stack two paper cups, press them together, hand them to the right arm, and place them into the rack.
Hang Towel	AC-One	Use one arm to hold the hanger and the other arm to place the towel onto it, then return the hanger to its position.

error accumulation. Together, these suites evaluate whether the policy can follow language instructions across spatial variation, object variation, goal variation, and long-horizon task composition. Table 24 summarizes the instruction templates and task counts for the four LIBERO suites.

Table 24: LIBERO task details. We list the language instruction templates and the total number of tasks per suite.

Suite	Language Instruction Templates	#Tasks
Spatial	pick up the *OBJ SPATIAL_REL* and place it on the *TARGET*	10
Object	pick up the *FOOD* and place it in the *CONTAINER*	10
Goal	open/close the *CONTAINER* open the *DRAWER* and put the *OBJ* inside put the *OBJ* on/in the *TARGET* push the *OBJ* to the *POSITION* of the *TARGET* turn on the *APPLIANCE*	10
Long-10	put both *OBJ1* and *OBJ2* in the *CONTAINER* turn on the *APPLIANCE* and put the *OBJ* on it put the *OBJ* in the *CONTAINER/APPLIANCE* and close it place *OBJ1* on *TARGET1* and *OBJ2* on *TARGET2*/at *REL* of *TARGET2* pick up the *OBJ* and place it in the caddy *COMPARTMENT*	10

E.3 RoboTwin 2.0 Tasks

The RoboTwin 2.0 benchmark evaluates bimanual manipulation in simulation with the Aloha-AgileX robot. We evaluate 13 tasks from the official benchmark, covering grasping, handover, lifting, articulated-object interaction, object placement, container manipulation, and bimanual stacking. RoboTwin 2.0 also provides Seen and UnSeen language settings. Seen instructions are sampled from the training prompt pool, while UnSeen instructions use held-out paraphrases that describe the same task with different wording. Table 25 shows one Open Laptop instruction from each split as a representative example. In addition to language variation, RoboTwin 2.0 is evaluated under Easy and Hard scene settings, where Easy corresponds to the clean ID setting and Hard corresponds to the randomized OOD setting.

On the six-task subset described in Appendix D.3, PriorVLA achieves 78.5% Easy and 48.5% Hard success with Seen instructions, compared with 77.3% and 48.3% with UnSeen instructions. FFT and LoRA show similarly small differences.

The full RoboTwin 2.0 task set used in our experiments contains the following 13 task names: *Stack Bowls Two*, *Stack Blocks Two*, *Put Object Cabinet*, *Put Bottles Dustbin*, *Place Phone Stand*, *Place Dual Shoes*, *Place Object Basket*, *Pick Dual Bottles*, *Open Laptop*, *Move Can Pot*, *Lift Pot*, *Handover Mic*, and *Grab Roller*.

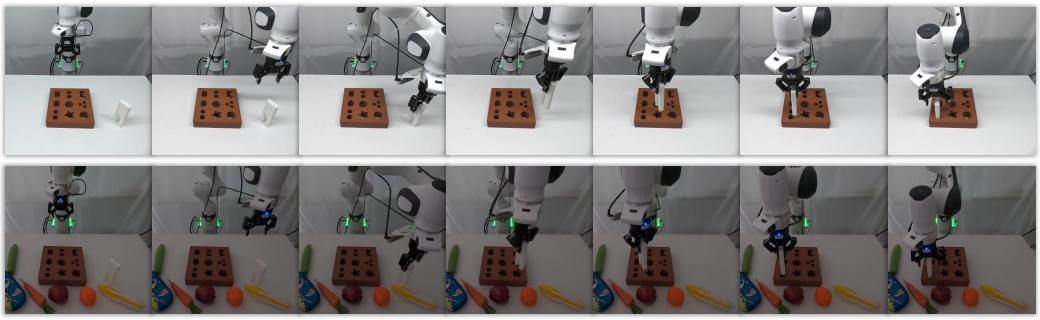
Table 25: RoboTwin 2.0 Open Laptop language-prompt example. We show one Seen instruction and one UnSeen instruction to illustrate the language split used in evaluation.

Language split	Example instruction
Seen	Lift the laptop with silver base open with the right arm.
UnSeen	Open the hinged screen laptop by lifting its lid.

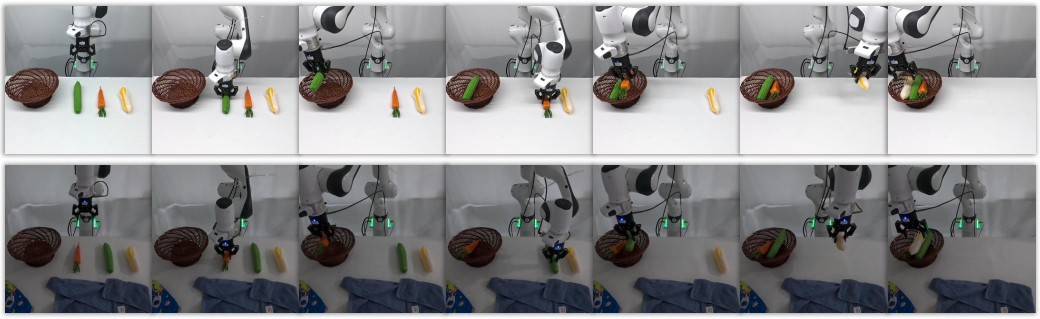
F Qualitative Results

We provide qualitative rollout examples to complement the quantitative results in the main paper. These visualizations illustrate representative policy executions in both real-world and simulation environments, and help show how the policy behaves under in-distribution (ID) and out-of-distribution (OOD) conditions. We first summarize what each qualitative figure contains, and then present the figures as a visual gallery.

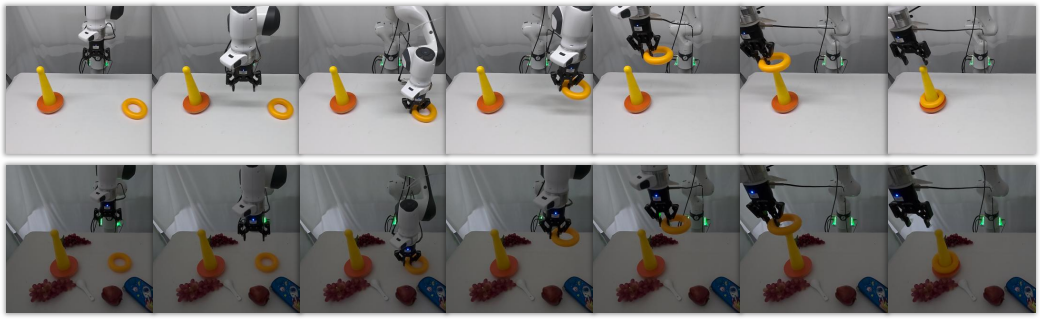
The qualitative rollout figures are organized as follows: (1) Fig. 6 shows Franka real-world rollouts across four tasks, with both ID and OOD trials. (2) Fig. 7 shows AC-One real-world rollouts across four dual-arm tasks, with both ID and OOD trials. (3) Fig. 8 shows the first set of RoboTwin 2.0 simulation rollouts under Easy/ID and Hard/OOD evaluation. (4) Fig. 9 shows additional RoboTwin 2.0 simulation rollouts under Easy/ID and Hard/OOD evaluation. (5) Fig. 10 shows representative LIBERO rollouts, one from each evaluated suite.



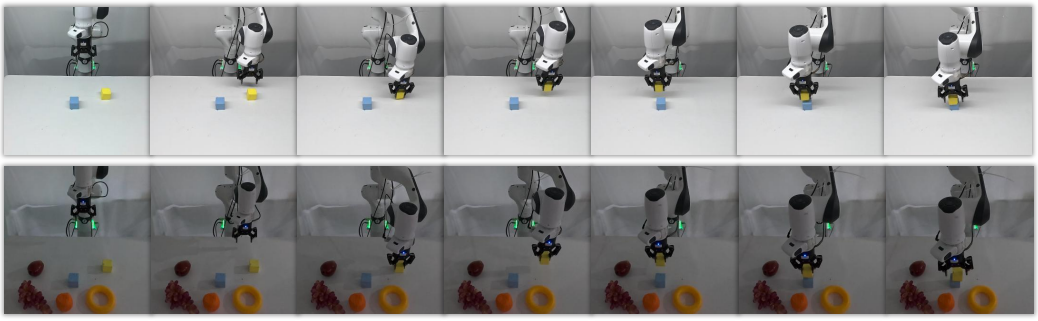
(a) Insert Peg



(b) Pick Object



(c) Place Ring

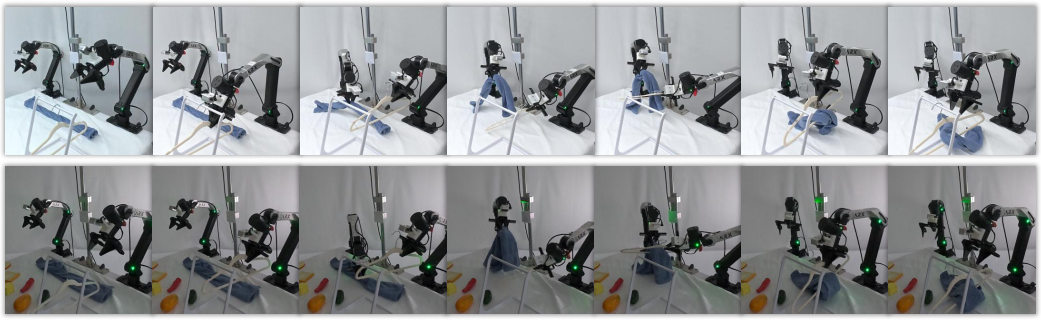


(d) Stack Blocks

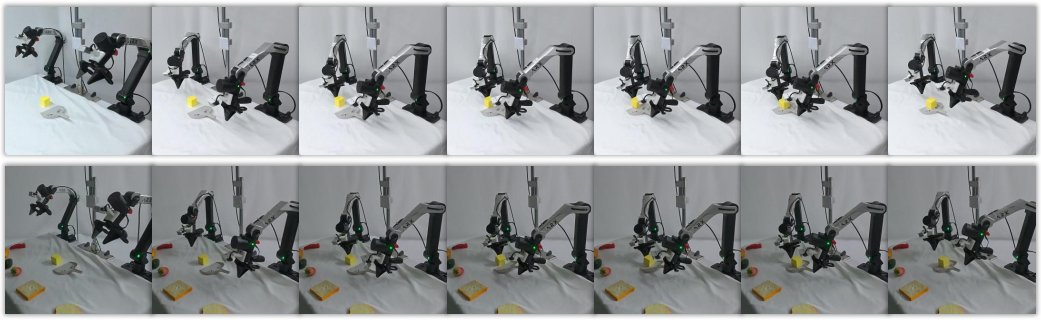
Figure 6: **Qualitative real-world results on the Franka platform.** We show rollout snapshots for four representative tasks under both in-distribution (ID) and out-of-distribution (OOD) evaluation. These examples illustrate representative task progression under nominal and perturbed real-world conditions.



(a) Arrange Cups



(b) Hang Towel



(c) Sweep Blocks

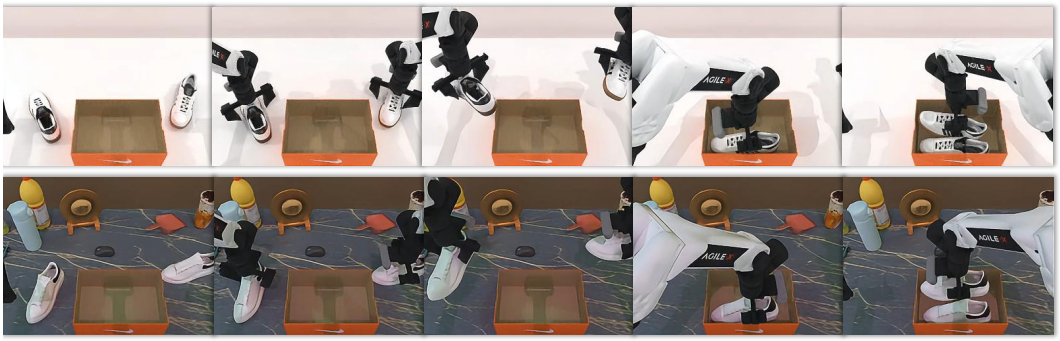


(d) Stack Bowls

Figure 7: **Qualitative real-world results on the AC-One platform.** We show rollout snapshots for four representative dual-arm tasks under both in-distribution (ID) and out-of-distribution (OOD) evaluation. These examples illustrate representative task progression on the dual-arm platform.



(a) Put Object Cabinet



(b) Place Dual Shoes

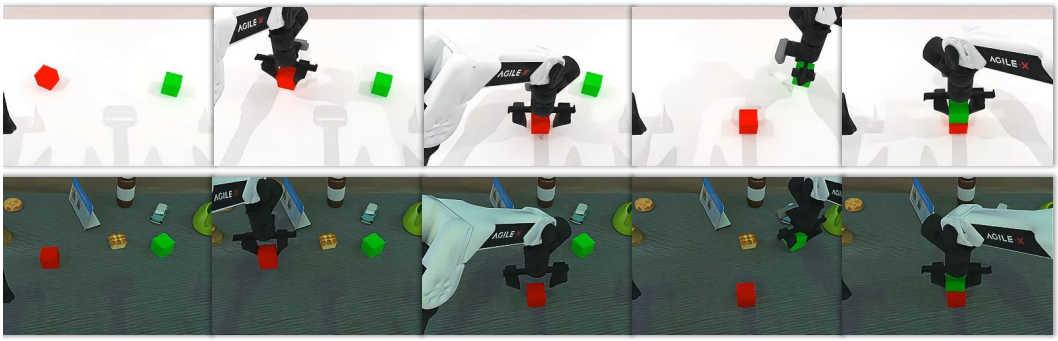


(c) Place Object Basket

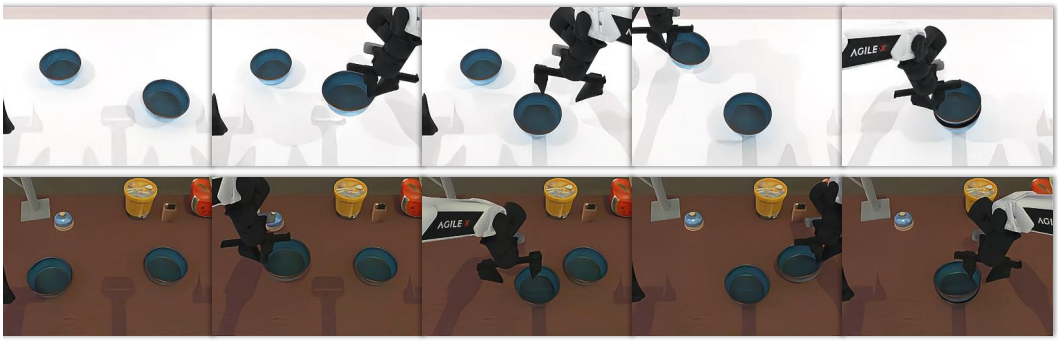
Figure 8: **Qualitative simulation results on RoboTwin 2.0 (Part I).** We show additional representative rollout snapshots under ID/Easy and OOD/Hard evaluation.



(d) Put Bottles Dustbin

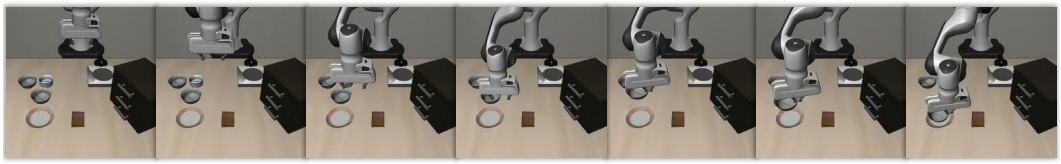


(e) Stack Blocks Two



(f) Stack Bowls Two

Figure 9: **Qualitative simulation results on RoboTwin 2.0 (Part II).** We show additional representative rollout snapshots under ID/Easy and OOD/Hard evaluation.



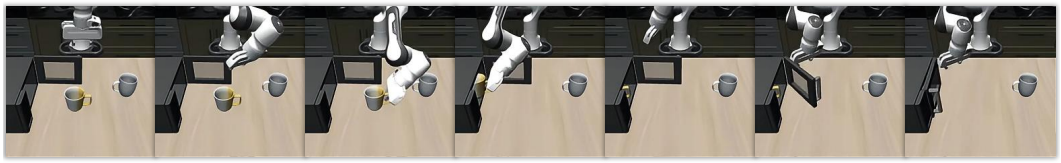
(a) Spatial: pick up the black bowl between the plate and the ramekin and place it on the plate



(b) Object: pick up the cream cheese and place it in the basket



(c) Goal: open the middle drawer of the cabinet



(d) Long: put the yellow and white mug in the microwave and close it

Figure 10: **Qualitative simulation results on LIBERO.** We show one representative task from each of the four suites. Since LIBERO is evaluated under the standard benchmark setting, only in-distribution rollouts are shown.

G VQA Analysis

To further examine whether downstream adaptation preserves visual-language reasoning ability, we conduct a small set of qualitative VQA probes. These probes are not used as an additional benchmark or training objective; instead, they provide diagnostic evidence about whether a model retains basic vision-language understanding after robot-task adaptation. The prompts cover both general visual understanding, such as object recognition, digit recognition, counting, and scene description, and robot-relevant questions, such as predicting the next subtask from the current manipulation scene.

For a fair comparison, we disable Scene Queries during this VQA probing stage. Scene Queries are a PriorVLA-specific interface designed to capture task-relevant scene priors for action generation. Keeping them active during VQA could introduce an additional query pathway that is not present in the pretrained model or the full fine-tuning baseline. We therefore evaluate all models through the shared vision-language pathway, so that the comparison reflects how much visual-language ability is preserved after adaptation rather than whether an additional query interface can assist VQA.

Figure 11 shows representative VQA results. The pretrained model produces coherent answers for all probes, confirming that the base VLA retains general visual-language understanding before downstream robot-task adaptation. Full fine-tuning, however, produces non-semantic and unreadable responses in these examples. This suggests that directly updating the full model on action-supervision data can disturb the language-generation behavior of the pretrained VLM, even when the downstream objective is not designed to modify VQA ability. In contrast, PriorVLA preserves coherent VQA responses after adaptation. It correctly recognizes common objects and digits, counts scene elements, generates a meaningful scene description, and produces reasonable robot-relevant subtask answers.

These results are consistent with the motivation of PriorVLA. Full fine-tuning uses the pretrained model primarily as an initialization for downstream action prediction, which can shift pretrained visual-language representations toward the narrow robot demonstration distribution. PriorVLA instead separates prior preservation from downstream specialization. The frozen Prior Expert preserves action-side motor priors, the VLM core is kept largely frozen while the vision encoder and adaptation branch specialize to downstream actions. As a result, PriorVLA can improve downstream robot performance while retaining more of the pretrained model's vision-language functionality. Although the VQA probes are qualitative and not intended as a standalone benchmark, they provide supporting evidence that PriorVLA better preserves general-purpose visual-language priors during robot-task adaptation.

H Additional Discussion

H.1 Why Prior Preservation Helps OOD Generalization

Out-of-distribution generalization in downstream VLA adaptation is challenging because downstream demonstrations often cover only a limited range of visual conditions, object configurations, and workspace layouts. Full fine-tuning updates the pretrained action-generation pathway directly with this restricted supervision. While this can improve performance on the training distribution, it may also shift pretrained representations and action priors toward narrow downstream patterns, reducing robustness when evaluation scenes contain unseen lighting, backgrounds, object poses, clutter, or workspace changes.

PriorVLA mitigates this issue by separating prior preservation from downstream specialization. The frozen Prior Expert preserves the pretrained action-generation pathway and serves as a read-only source of motor-prior representations. The Adaptation Expert, initialized from the same pretrained action expert, specializes to downstream action generation. Expert Queries then provide learnable interfaces for using preserved priors: Scene Queries capture task-relevant scene priors from the VLM, Motor Queries capture motor-prior representations from the frozen Prior Expert, and Action Queries integrate these priors inside the trainable Adaptation Expert.

This design is especially useful under distribution shift. When visual scenes or object configurations differ from the downstream training data, the policy cannot rely only on patterns observed in task-specific demonstrations. It can instead reuse priors learned during large-scale pretraining, such as object-centric visual grounding, affordance-related scene understanding, and general motor regular-

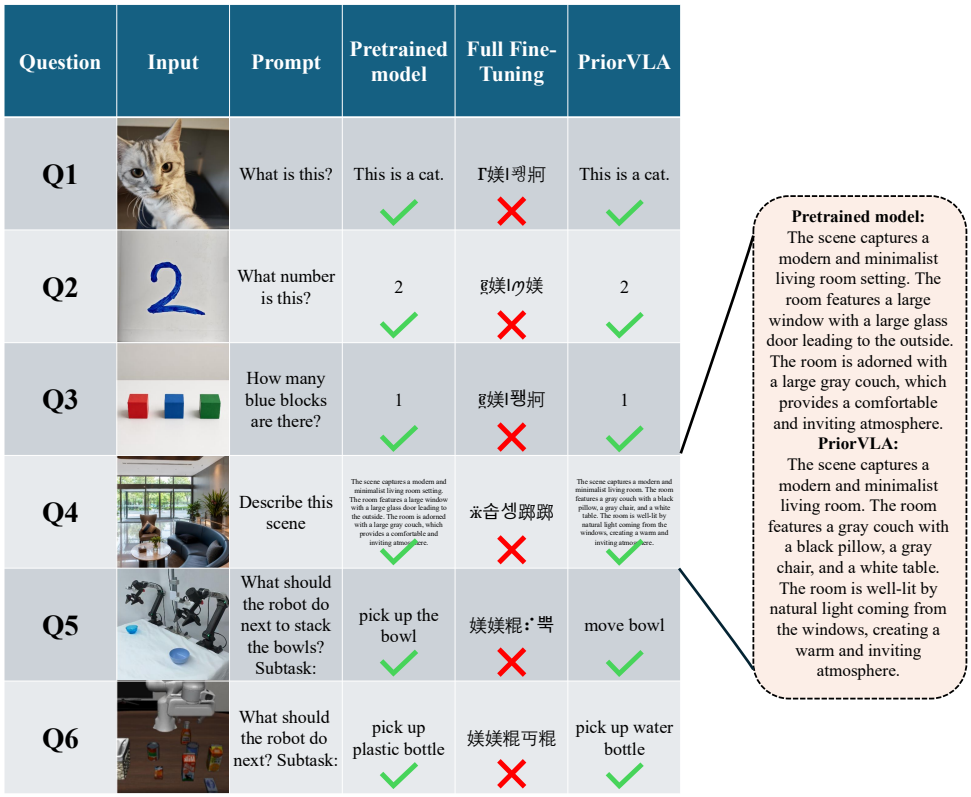


Figure 11: Qualitative VQA examples after adaptation. The probes include general visual recognition, counting, scene description, and robot-relevant subtask questions. During VQA probing, Scene Queries are disabled to ensure that PriorVLA does not use an additional VQA-specific query pathway. Full fine-tuning produces non-semantic outputs in these examples, while PriorVLA preserves coherent visual-language responses similar to the pretrained model.

ities. The RoboTwin 2.0-Hard and real-world OOD results are consistent with this interpretation: PriorVLA shows larger gains under OOD evaluation than under easier ID settings.

H.2 Why Prior Preservation Helps Few-Shot Adaptation

Few-shot adaptation further amplifies the risk of over-specialization because a small number of downstream demonstrations covers only a narrow subset of initial states, visual appearances, object poses, and successful action trajectories. In this regime, full fine-tuning can make the adapted policy sensitive to incidental correlations in the few-shot data, rather than learning task-relevant structure that transfers across evaluation conditions.

PriorVLA reduces this risk by separating a frozen prior source from a trainable adaptation branch. The frozen Prior Expert maintains a stable source of motor-prior features, while the Adaptation Expert learns the downstream action mapping. Expert Queries serve as learnable interfaces for selecting and integrating preserved priors, rather than replacing the pretrained prior itself. As a result, the model does not need to recover all relevant visual and motor structure from few-shot data alone; it can reuse pretrained priors and learn how to integrate them for the downstream task.

The few-shot results are consistent with this view. PriorVLA improves over full fine-tuning in both simulation and real-world few-shot settings, with especially strong gains under OOD evaluation. This suggests that the advantage of PriorVLA comes not only from parameter efficiency, but also from preserving and reusing pretrained knowledge when downstream supervision is limited.

H.3 Broader Impacts

PriorVLA aims to improve data-efficient adaptation of pretrained robot policies. Its potential positive impact is to reduce the amount of task-specific robot data needed for downstream adaptation, which may lower the cost of developing robot systems and improve robustness under moderate distribution shifts.

Potential negative impacts may arise if stronger robot policies are deployed without sufficient validation or supervision. Failures under unseen scenes, contact-rich interactions, or unsafe environments could lead to physical damage or unintended robot behavior. The method could also be misused in inappropriate or unauthorized automation settings if deployed without task and environment restrictions. In our work, real-world experiments are conducted in controlled lab settings with human supervision, task-specific evaluation, and standard safety procedures. We do not claim that the policy is ready for unsupervised deployment.

H.4 Real-World Deployment Notes

Real-robot experiments are conducted under human supervision and with standard safety procedures. A trial is counted as a failure if the robot requires human intervention, triggers a safety stop, violates the task constraints, or fails to satisfy the success criterion before timeout. These criteria are applied consistently across methods.

During deployment, the policy predicts an action chunk of length 50 and executes the first 15 actions before replanning with a new observation. This closed-loop chunking strategy limits open-loop drift while keeping the inference frequency manageable for real-robot control. The same action-horizon and execution-horizon settings are used across the evaluated real-robot tasks.

Before each evaluation batch, the workspace is reset according to the task protocol. We check workspace boundaries, robot speed limits, emergency-stop access, camera visibility, object placement, and table clearance. For OOD evaluation, perturbations such as lighting changes, background-object changes, initial object-pose variation, and table-height changes are applied while maintaining safe operating conditions. These procedures ensure that failures are attributed to policy behavior under the specified evaluation condition rather than to uncontrolled hardware or reset inconsistencies.

H.5 Existing Assets and Licenses

We build on and evaluate with existing open-source assets, and follow their corresponding licenses and terms of use. Table 26 summarizes the main external assets used in this work.

Table 26: **Existing assets used in PriorVLA.** We cite the corresponding papers and respect the licenses and terms of use of each asset.

Asset	**Use in this work**	**License / Terms**
OpenPI ($\pi_{0.5}$)	Base VLA implementation and pretrained backbone used for adaptation.	Apache-2.0 for the OpenPI codebase; model/checkpoint use follows the upstream terms, including the Gemma Terms of Use where applicable.
RoboTwin 2.0	Simulation benchmark, official demonstrations, and data-generation pipeline for few-shot and large-data regimes.	MIT License.
LIBERO	Simulation benchmark and public task demonstrations for the four LIBERO suites.	MIT License for code; CC BY 4.0 for datasets.
LeRobot	LeRobot data format and tooling used for converting and organizing demonstrations.	Apache-2.0 License.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main contributions and experimental scope, including the PriorVLA framework, its prior-preserving design, and evaluations on RoboTwin2, LIBERO, and real-world tasks. These claims are supported by the method in Sec. 3, experiments in Sec. 4, and additional results in Appendix D.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 5, Appendix C, and Appendix H.4 discuss the scope and deployment considerations of PriorVLA, including inference computation, the role of pretrained VLA coverage, and the finite scope of evaluated tasks, embodiments, and deployment environments.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present theoretical results, theorems, or formal proofs. The equations in Sec. 3 define the problem formulation, model architecture, and training objective.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the main implementation details, training data construction, hyperparameters, training procedure, checkpointing rules, and evaluation protocols in Appendix A, Appendix B, and Appendix C. We also report full per-task results and consistency analysis in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provided anonymized source code, configuration files, environment setup, and training/evaluation commands in the supplemental material. RoboTwin2 experiments use the official open-source demonstrations and the official RoboTwin2 data-generation pipeline for additional data regimes, while LIBERO uses public benchmark data. For real-world experiments, Appendix B.1 and Appendix C document the platform setup, camera configuration, task suite, training details, and evaluation protocol. The code is publicly available at `https://github.com/xinyuguo1566/PriorVLA`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the training data, data regimes, optimizer, hyperparameters, grouped learning-rate multipliers, checkpointing rules, and compute resources in Appendix B. We describe the RoboTwin2, LIBERO, and real-world evaluation protocols, including trial counts, ID/OOD settings, and success criteria, in Appendix C.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Appendix D.2 reports task-setting consistency and one-sided exact binomial sign tests against $\pi_{0.5}$. Appendix D.6 reports mean success rates and sample standard deviations across three independent training seeds on the six-task RoboTwin2 subset. The evaluation trial counts for RoboTwin2, LIBERO, and real-world experiments are specified in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Appendix B.3 reports the GPU type, number of GPUs, memory capacity, global batch size, training steps, and representative wall-clock time per trained model for the main experimental settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms to the NeurIPS Code of Ethics. It uses public simulation benchmarks and controlled real-world robot experiments, does not involve human-subject studies or privacy-sensitive data, and the submission was anonymized for peer review.

Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Appendix H.3 discusses potential positive impacts, including more data-efficient and robust robot adaptation, as well as potential negative impacts from unsafe or unauthorized deployment. It also describes the need for controlled evaluation and supervised deployment.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models, scraped datasets, or directly deployable robot policy checkpoints. The supplemental material provided anonymized source code and configuration files for research reproducibility, while real-world experiments are documented as controlled robotics evaluations.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers and repositories for the external assets used in this work, including OpenPI/$\pi_{0.5}$, RoboTwin2, LIBERO, and LeRobot-style data formatting. Appendix H.5 summarizes the main assets, their use in this work, and their licenses or terms of use.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We provided anonymized source code, configuration files, and documentation in the supplemental material. The documentation describes the environment setup, data preparation for public benchmarks, and training/evaluation commands.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human-subject experiments. Real-world demonstrations are collected for robot policy training and evaluation, and the work does not analyze human participants, personal data, or participant behavior.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human-subject research. Real-world demonstrations are collected for robot policy training and evaluation, and the work does not study human participants, personal data, or participant behavior.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development does not use external LLMs as important, original, or non-standard components. Any LLM use was limited to writing, editing, or formatting and did not affect the methodology, experiments, or scientific conclusions.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.