arXiv:2608.15285v1 [cs.RO] 15 Aug 2026

PhaseLoRA: Control-Regime-Conditioned Low-Rank Adaptation for Continuous-Action Vision-Language-Action Policies

Yufei Guo
Tsinghua University
guoyufei21@mails.tsinghua.edu.cn

Yinan Wu
Tsinghua University
yinanwu@tsinghua.edu.cn

Haoran Duan
Tsinghua University
haoranduan@tsinghua.edu.cn

Guiguang Ding
Tsinghua University
dinggg@tsinghua.edu.cn

Jungong Han*
Tsinghua University
jghan@mail.tsinghua.edu.cn

Abstract

Parameter-efficient fine-tuning (PEFT) is a natural way to adapt pretrained vision-language-action (VLA) policies, but most adapter designs apply temporally static updates throughout a control rollout, overlooking the phase-dependent nature of continuous-action manipulation. Such policies traverse distinct regimes, including approach, contact transition, grasping, transport, and placement, each requiring different adaptation behaviors. We propose **PhaseLoRA**, a lightweight LoRA parameterization that conditions adaptation at each action-chunk prediction step using two weakly supervised descriptors: fine-control tendency and event/boundary intensity. PhaseLoRA modulates the LoRA left factor in the action expert, allowing the effective low-rank update direction to vary over time while keeping the backbone largely frozen. On LIBERO, PhaseLoRA improves average success rate by 12.2 points over a matched-parameter high-rank LoRA baseline and outperforms stronger LoRA variants. Ablations show that random temporal modulation and scalar gating do not reproduce the performance of the full model, while update-direction analyses reveal structured temporal variation associated with the predicted control descriptors. These results establish within-trajectory conditioning as an effective lightweight PEFT axis for continuous-action VLA policies.

Code: The code will be released at `https://github.com/Grinffin/PhaseLoRA`.

1 Introduction

Pretrained vision-language-action (VLA) policies unify visual perception, language conditioning, and action generation, making them strong starting points for robot manipulation [2, 6, 37, 27, 26, 19, 1]. As these backbones grow in scale, downstream adaptation increasingly relies on parameter-efficient fine-tuning (PEFT), especially low-rank adaptation (LoRA) [10], to avoid the cost of full fine-tuning [19, 18]. However, most PEFT methods treat adaptation as static within an execution: once

*Corresponding author.

Preprint.

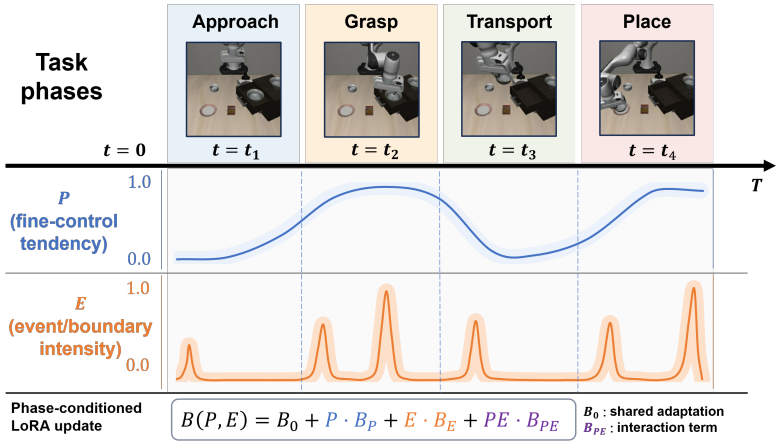


Figure 1: **Motivation for within-trajectory control-regime conditioning.** A single manipulation rollout can traverse different control regimes, including approach, grasp, transport, and precise placement. We summarize these variations with two continuous descriptors: fine-control tendency P and event/boundary intensity E. PhaseLoRA uses these descriptors to modulate the LoRA left factor, $B(P, E) = B_0 + P \cdot B_P + E \cdot B_E + PE \cdot B_{PE}$, so the effective low-rank update can vary across control steps instead of remaining static throughout the rollout.

an adapter is chosen for a task, instruction, or example [14, 25, 16], the same update is applied throughout the control rollout.

This temporal invariance overlooks a basic property of continuous-action manipulation: a single rollout is not a homogeneous control problem. It may progress through coarse approach, contact or release transitions, grasp execution, transport, and precise placement, as also reflected in skill-structured and contact-rich manipulation studies [30, 5, 8, 34, 36]. These regimes can require different corrections to the pretrained policy even within the same task and instruction. We therefore view within-trajectory control heterogeneity as a missing adaptation axis for PEFT in continuous-action VLA policies: the adapter should remain lightweight, but its effective update should be allowed to change with the evolving control regime.

We introduce **PhaseLoRA**, a parameter-efficient adapter that conditions LoRA updates at each action-chunk prediction step using two weakly supervised descriptors: *fine-control tendency* and *event/boundary intensity*. As illustrated schematically in Figure 1, these descriptors are designed to represent precision-demanding moments, such as grasping and placement, and abrupt control changes, such as contact or release. Unlike conditional LoRA methods that adapt across tasks, examples, experts, or instructions [14, 25, 16, 33], PhaseLoRA conditions adaptation within a single rollout. It modulates the LoRA left factor with shared, descriptor-specific, and interaction components, letting the effective low-rank update direction vary over time while keeping the backbone largely frozen. The descriptors are learned from trajectory statistics, without manual phase labels or full-capacity expert branches.

We evaluate whether within-trajectory conditioning provides benefits beyond standard explanations such as increased parameter count or generic temporal variation. On LIBERO [22], PhaseLoRA consistently improves over standard LoRA, a matched-parameter high-rank LoRA baseline, and stronger LoRA variants. Through ablations and update-direction analyses, we test whether control-informed descriptors outperform random temporal modulation, whether scalar rescaling of a fixed LoRA direction is sufficient, and whether the learned low-rank update directions vary near salient manipulation events. We further evaluate PhaseLoRA on real-world manipulation tasks, providing supporting evidence that the observed benefits extend beyond simulation to physical execution.

Our contributions are:

- We formulate within-trajectory control heterogeneity as a missing adaptation axis for PEFT in continuous-action VLA policies.

- We introduce **PhaseLoRA**, a lightweight LoRA parameterization that conditions the adapter on weakly supervised fine-control and event/boundary descriptors while preserving low-rank updates at each action-chunk prediction step.
- We show that PhaseLoRA outperforms the matched-parameter high-rank LoRA, random-modulation, and scalar-gating controls, and analyze how its effective low-rank update directions vary with the predicted descriptors and near salient manipulation events.

2 Related Work

Generalist robot policies and vision-language-action models. Generalist robot learning has advanced from multitask imitation policies such as BC-Z [15] and PerAct [28] to large-scale robot foundation models trained with multimodal and multi-embodiment data. RT-1 [2] showed that scaling data and model capacity can improve real-world robotic control, while PaLM-E [6] and RT-2 [37] connected pretrained language or vision-language models to embodied action generation. Recent systems such as RT-X [27], Octo [26], OpenVLA [19], $\pi_{0.5}$ [1], and RDT-1B [24] further establish pretrained VLA backbones as practical starting points for robot manipulation. Our work builds on this setting and focuses on how to adapt such continuous-action VLA policies efficiently to downstream manipulation tasks.

Fine-tuning and parameter-efficient adaptation for robot foundation models. As robot foundation models scale, efficient downstream adaptation has become a practical bottleneck. OpenVLA [19] identified efficient fine-tuning as important for VLA deployment, and OpenVLA-OFT [18] systematically studied adaptation choices such as decoding, action representation, action chunking, and training objectives. Parameter-efficient VLA adaptation has also been explored through LoRA-style tuning, input- and layer-wise adaptive rank allocation [17], and sequential LoRA adaptation in continual RL [11]. RL-based post-training methods such as CO-RFT [12] and VLA-RFT [21] are complementary to our setting, as they mainly modify the learning signal rather than the adapter parameterization. Our focus is also complementary: rather than adapting rank or the post-training objective, we condition the LoRA update direction on weakly supervised control descriptors within a single rollout for continuous-action VLA fine-tuning.

LoRA variants, dynamic adapters, and conditional parameter generation. Beyond robotics, LoRA has been extended through rank or budget allocation, such as DyLoRA [29] and AdaLoRA [35], alternative parameterizations such as DoRA [23], and mixtures of LoRA experts such as X-LoRA [3] and D-MoLE [33]. Structured PEFT methods further show that a single shared low-rank update can be limiting under heterogeneous adaptation demands: MokA [32] separates unimodal and cross-modal adaptation, while HINT [14], HyperLoRA [25], and Conditional LoRA Parameter Generation [16] generate adapter parameters conditionally. Our distinction is within-rollout conditioning at each policy query for continuous-action VLA policies. Prior conditional LoRA methods mainly adapt across tasks, modalities, instructions, experts, or input examples; PhaseLoRA instead modulates the action-expert LoRA update once per action-chunk prediction within a single rollout, using weakly supervised descriptors of the current control regime. This targets temporal control heterogeneity in manipulation without requiring explicit phase labels or full-capacity expert branches.

Phase structure, skill decomposition, and contact-rich manipulation. Another line of work observes that manipulation is rarely a homogeneous control process. Long-horizon imitation methods often decompose demonstrations into skills or introduce intermediate representations, as in The Art of Imitation [30], TacUMI [5], and RoboInter [20]. Contact-rich manipulation methods model interaction regimes more explicitly, for example by separating reaching from local interaction [36] or conditioning policies on force or tactile information, as in TLA [8], ForceVLA [34], Tactile-VLA [13], and HapticVLA [7]. Concurrent VLA methods have also introduced phase-conditioned action generation: BehaviorVLA uses a phase-conditioned behavior decoder, while Mag-VLA combines a motion-aware phase classifier with a phase-conditioned Action Chunking Transformer decoder [9, 31]. These works motivate our view that a single rollout can contain heterogeneous control regimes. We differ by conditioning the low-rank adapter rather than the action decoder, without requiring explicit skill boundaries, manual phase labels, or force or tactile measurements.

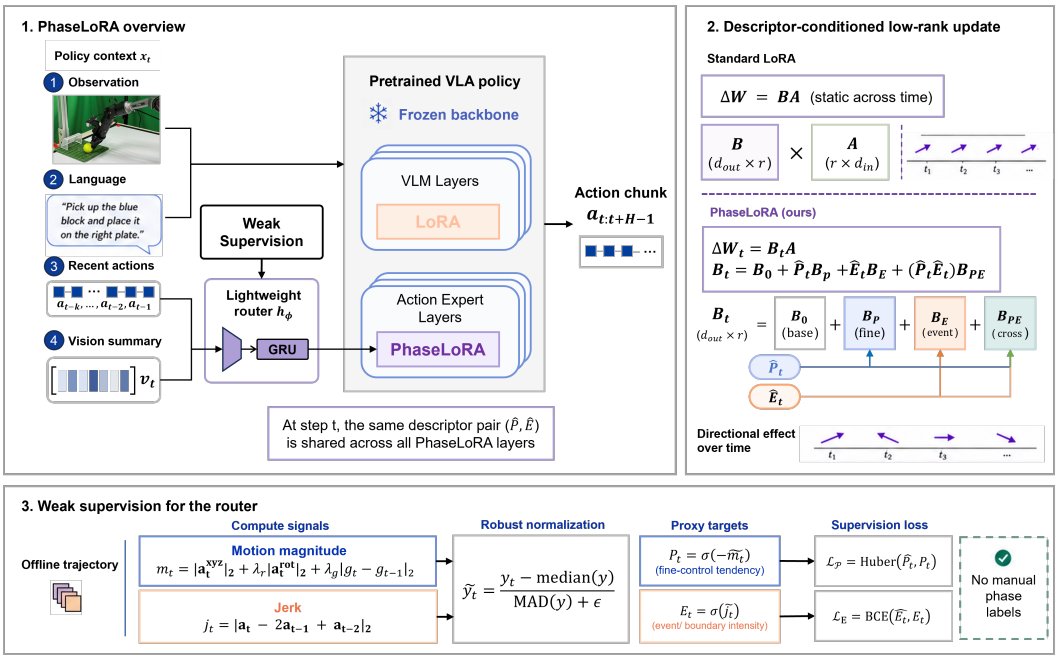


Figure 2: **Overview of PhaseLoRA.** A lightweight router predicts fine-control tendency $\hat{P}_t$ and event/boundary intensity $\hat{E}_t$ from the current policy context. These descriptors modulate the action-expert LoRA update through shared, descriptor-specific, and interaction components, yielding control-regime-dependent low-rank corrections within a rollout. The router is trained with weak proxy targets derived from trajectory statistics.

3 Method

3.1 Setup

We study parameter-efficient fine-tuning for continuous-action vision-language-action (VLA) policies. Let t denote a low-level control timestep at which the policy is queried, and let x_t denote the policy context available at that query, including the observation, language instruction, and any additional state inputs used by the backbone. A pretrained VLA policy predicts an action chunk

$$\mathbf{a}_{t:t+H-1} = f_\theta(x_t), \tag{1}$$

where H is the action horizon and θ denotes the pretrained policy parameters.

We consider low-rank adaptation while keeping the pretrained backbone frozen except for a small set of trainable adaptation parameters. For a target linear layer with pretrained weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, standard LoRA replaces W with

$$W' = W + \Delta W, \qquad \Delta W = BA, \tag{2}$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A \in \mathbb{R}^{r \times d_{\text{in}}}$ are trainable low-rank factors, and r is the adaptation rank. For clarity, we omit the standard LoRA scaling factor in notation.

Standard LoRA applies the same update matrix ΔW at every policy query. This is restrictive for continuous-action manipulation, where a single rollout can traverse substantially different control regimes, from coarse motion and contact transitions to grasp execution and precise alignment. Our goal is therefore to replace the static update with a within-trajectory, control-regime-conditioned low-rank adaptation while retaining the efficiency of LoRA.

3.2 PhaseLoRA

Our key idea is to make the LoRA update depend on the current control regime within a trajectory. At a policy query issued at timestep t, a lightweight router predicts two scalar control descriptors

$(\hat{P}_t, \hat{E}_t) \in [0,1]^2$ for the forward pass that produces the action chunk $\mathbf{a}_{t:t+H-1}$. The same descriptor pair is shared across all PhaseLoRA layers for that forward pass, while each layer keeps its own low-rank parameters.

We apply descriptor-conditioned adaptation only to the policy's action-prediction module, referred to as the *action expert*; any VLM-side LoRA parameters remain standard and time-invariant. For an adapted action-expert layer with pretrained weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, PhaseLoRA replaces the static LoRA update with

$$\Delta W_t = B_t A, \tag{3}$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$ is the trainable right factor for that layer and is shared across policy queries. The left factor is conditioned on the predicted descriptors:

$$B_t = B_0 + \hat{P}_t B_P + \hat{E}_t B_E + (\hat{P}_t \hat{E}_t) B_{PE}, \tag{4}$$

with $B_0, B_P, B_E, B_{PE} \in \mathbb{R}^{d_{\text{out}} \times r}$.

The shared component B_0 parameterizes the time-invariant part of the adapter, while B_P, B_E, and B_{PE} parameterize descriptor-dependent deviations associated with fine-control tendency, event/boundary intensity, and their interaction. We condition only the left factor: the shared right factor A defines a common input projection, whereas the descriptor-conditioned B_t changes the output directions of the low-rank correction. For any fixed $(\hat{P}_t, \hat{E}_t)$, the update remains rank at most r, since ΔW_t is still the product of a $d_{\text{out}} \times r$ matrix and an $r \times d_{\text{in}}$ matrix. Thus, PhaseLoRA preserves the low-rank structure of LoRA while allowing the effective update direction to vary across policy queries, rather than only rescaling a fixed update.

3.3 Weak Supervision for Control Descriptors

The control descriptors are not directly observed. Instead, during training, we construct weak proxy targets for two interpretable quantities: a fine-control tendency score $P_t \in [0,1]$ and an event/boundary intensity score $E_t \in [0,1]$. These targets are computed offline from the low-level action and gripper-command sequence of each training trajectory, and used only to supervise the router.

Fine-control tendency. We define a motion statistic

$$m_t = \|\mathbf{a}_t^{xyz}\|_2 + \lambda_r \|\mathbf{a}_t^{rot}\|_2 + \lambda_g |g_t - g_{t-1}|, \tag{5}$$

where $\mathbf{a}_t^{xyz}$ and $\mathbf{a}_t^{rot}$ are the translational and rotational components of the low-level action at step t, g_t is the gripper command, and λ_r, λ_g are fixed coefficients used to balance terms with different scales. Lower motion magnitude is treated as a weak indicator of finer-grained control.

Event/boundary intensity. To capture abrupt control changes, we define

$$j_t = \|\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\|_2, \tag{6}$$

where j_t measures action jerk.

For each trajectory, we robustly normalize each statistic over time. For $y \in \{m, j\}$, let y also denote the corresponding per-trajectory time series; we define

$$\tilde{y}_t = \frac{y_t - \text{median}(y)}{\text{MAD}(y) + \epsilon}, \tag{7}$$

where $\text{MAD}(\cdot)$ denotes the median absolute deviation. We then form the weak proxy targets as

$$P_t = \sigma(-\tilde{m}_t), \qquad E_t = \sigma(\tilde{j}_t), \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function.

We emphasize that P_t and E_t are weak proxy targets, not exact symbolic phase labels. Their purpose is not to recover a unique segmentation of the task, but to provide low-dimensional supervision for learning continuous control descriptors that approximately track fine-control tendency and event intensity.

3.4 Router and Training

Since the weak proxy targets (P_t, E_t) are unavailable at inference time, we train a lightweight router h_ϕ to predict descriptors from information available up to control step t:

$$(\hat{P}_t, \hat{E}_t) = h_\phi(\mathbf{z}_t), \qquad \hat{P}_t, \hat{E}_t \in [0, 1]. \tag{9}$$

The router input $\mathbf{z}_t$ includes a visual-semantic summary from the VLA backbone, recent action and gripper history. The history branch uses the previous $K = 6$ control steps and retains the first 5 executed low-level actions from each past action chunk. During training, the history is taken from offline past actions; during inference, it is updated online from previously executed model outputs. The router is a one-layer GRU with hidden dimension 128 followed by two sigmoid heads; further details are in the appendix.

We jointly optimize PhaseLoRA and the router. Let $\mathcal{L}_{\text{policy}}$ be the original VLA action-prediction loss. The router is supervised by the weak targets from Sec. 3.3:

$$\mathcal{L}_P = \text{Huber}(\hat{P}_t, P_t), \qquad \mathcal{L}_E = \text{BCE}(\hat{E}_t, E_t), \tag{10}$$

with losses averaged over valid control steps. The total objective is

$$\mathcal{L} = \mathcal{L}_{\text{policy}} + \lambda_P \mathcal{L}_P + \lambda_E \mathcal{L}_E + \mathcal{L}_{\text{reg}}, \tag{11}$$

where λ_P and λ_E weight the auxiliary supervision terms. We regularize descriptor-dependent LoRA components across PhaseLoRA layers:

$$\mathcal{L}_{\text{reg}} = \sum_{\ell \in \mathcal{S}} \left(\lambda_P^{\text{reg}} \|B_{P,\ell}\|_F^2 + \lambda_E^{\text{reg}} \|B_{E,\ell}\|_F^2 + \lambda_{PE}^{\text{reg}} \|B_{PE,\ell}\|_F^2 \right), \tag{12}$$

where $\mathcal{S}$ is the set of PhaseLoRA layers. We initialize B_P, B_E, and B_{PE} to zero and initialize (B_0, A) with the standard LoRA initialization, so each layer starts as ordinary LoRA and learns descriptor-dependent deviations during fine-tuning.

3.5 Complexity

Compared with standard LoRA, our method introduces additional left factors while keeping the right factor fixed across control steps. For a layer with input dimension d_{in}, output dimension d_{out}, and rank r, standard LoRA uses $r(d_{\text{in}} + d_{\text{out}})$ trainable parameters. Our method uses $rd_{\text{in}} + 4rd_{\text{out}}$, corresponding to one right factor A and four left factors (B_0, B_P, B_E, B_{PE}). Relative to standard LoRA, the additional parameter cost is therefore $3rd_{\text{out}}$ per adapted layer, plus the lightweight router.

At inference time, the extra computation comes from (i) evaluating the router and (ii) computing the descriptor-conditioned left factor in Eq. (4). No additional full-capacity expert branches are introduced. Thus, relative to standard LoRA, our method adds only a small descriptor-dependent overhead, while remaining substantially more efficient than full fine-tuning.

4 Experiments

We evaluate PhaseLoRA with four goals: whether within-trajectory conditioning improves over static LoRA, whether the gain can be explained by increased adaptation capacity, whether descriptor-conditioned updates are necessary, and whether the learned descriptors align with their intended control roles.

4.1 Experimental Setup

Benchmarks. Our primary controlled benchmark is LIBERO [22], where we report the main comparisons, ablations, and descriptor analyses. We evaluate on the standard LIBERO suites and report both suite-level success rates and their macro average. Unless otherwise stated, ablations and descriptor analyses are conducted on LIBERO.

Table 1: Main results on LIBERO. We report success rates (%) on the four standard LIBERO suites and their overall average, together with the number of trainable parameters for each PEFT method. Results are reported as mean ± standard deviation over 3 random seeds.

Method	#Params	LIBERO(%)				
		Spatial	Object	Goal	10	Avg.
LoRA [10]	2.70%	60.7 ± 0.6	41.3 ± 2.1	32.7 ± 1.5	18.7 ± 0.6	38.3 ± 1.2
High-rank LoRA	4.42%	71.0 ± 1.0	73.3 ± 1.2	55.3 ± 3.1	27.0 ± 4.6	56.7 ± 2.3
DoRA [23]	4.42%	72.7 ± 0.6	74.0 ± 2.6	57.3 ± 2.5	29.7 ± 3.5	58.4 ± 2.2
LoRA-MoE [33]	4.47%	73.3 ± 1.5	70.7 ± 2.5	58.0 ± 2.0	31.0 ± 4.6	58.3 ± 2.4
LoRA-SP [17]	4.42%	76.3 ± 3.2	72.0 ± 2.0	56.0 ± 1.0	32.3 ± 2.5	59.2 ± 2.3
Ours	4.41%	$\mathbf{85.3 \pm 1.2}$	$\mathbf{88.0 \pm 2.0}$	$\mathbf{64.0 \pm 2.0}$	$\mathbf{38.3 \pm 3.2}$	$\mathbf{68.9 \pm 2.4}$

Real-world benchmark. We additionally evaluate on a four-task real-world tabletop manipulation benchmark collected with a Piper robotic arm and trained using the LeRobot [4] framework. The tasks require moving a tennis ball from a yellow plate to a blue plate, removing a cuboid from a blue plate, picking a red cube into a yellow plate, and putting an apple on a yellow plate. Each task provides 100 demonstrations for fine-tuning, and each method is evaluated with 10 trials per task for each of 3 random seeds, resulting in 30 trials per task per method. We report the mean and standard deviation across the 3 seeds.

Backbones and adaptation setting. Unless otherwise stated, our main experiments use the pre-trained $\pi_{0.5}$ backbone. Within each backbone, all methods use the same observation and language inputs, action representation, action chunk length, training data, and adapted module family. To isolate the effect of the adaptation mechanism, each PEFT method is applied to the same subset of layers. For PhaseLoRA, descriptor-conditioned adaptation is applied only to the action expert, following Sec. 3.2; VLM-side LoRA parameters, when present, remain standard and time-invariant.

Compared methods. We compare PhaseLoRA against five PEFT baselines: **LoRA** [10], standard low-rank adaptation with a static update; **High-rank LoRA**, a larger-rank baseline chosen to approximately match PhaseLoRA's trainable-parameter budget; **DoRA** [23], a weight-decomposed low-rank adaptation baseline; **LoRA-MoE** [33], a mixture-based low-rank adaptation baseline; and **LoRA-SP** [17], a structured LoRA baseline for VLA fine-tuning. High-rank LoRA controls for trainable capacity, while DoRA, LoRA-MoE, and LoRA-SP test whether stronger static or structured adapter variants recover the same gains.

Training and metrics. For each benchmark and backbone, all methods are trained with the same optimizer, batch size, data mixture, and number of training steps. Unless otherwise stated, LIBERO results are averaged over three random seeds, and real-world results are averaged over repeated trials for each task. We report episode-level success rate and the percentage of trainable parameters for each PEFT method. Additional implementation details are provided in the appendix.

4.2 Main Results on LIBERO

Table 1 reports the main LIBERO comparison. PhaseLoRA achieves the highest macro-average success rate, improving over standard LoRA by 30.6 points and over matched-parameter High-rank LoRA by 12.2 points. The comparison to standard LoRA tests whether a static low-rank update is sufficient, while the comparison to High-rank LoRA controls for trainable adaptation capacity. PhaseLoRA also outperforms DoRA, LoRA-MoE, and LoRA-SP, suggesting that stronger static or structured LoRA variants do not fully recover the benefit of within-trajectory control-regime conditioning.

4.3 Real-World Evaluation

We further evaluate PhaseLoRA on a real-world manipulation benchmark collected on our robot platform. Because physical evaluation is costly, we compare against the most relevant baseline: matched-parameter High-rank LoRA. This setting tests whether the trend observed on LIBERO also

Table 2: Main results on the real-world benchmark. We report mean $\pm$ standard deviation across 3 seeds, with 10 trials per task for each seed.

Method	#Params	Move	Remove	Pick	Put	Avg.
High-rank LoRA	4.42%	60.0 ± 8.2	53.3 ± 4.8	63.3 ± 9.4	26.7 ± 4.7	50.8 ± 4.2
Ours	4.41%	$\mathbf{66.7 \pm 4.8}$	$\mathbf{70.0 \pm 8.2}$	$\mathbf{90.0 \pm 8.2}$	$\mathbf{50.0 \pm 8.2}$	$\mathbf{69.2 \pm 3.1}$

appears under physical execution, and whether any improvement is explained solely by increased adaptation capacity.

Table 2 reports per-task and average success rates. PhaseLoRA achieves the highest average success rate, improving over High-rank LoRA by 18.4 points. These results provide supporting evidence that within-trajectory descriptor-conditioned adaptation can improve continuous-action VLA fine-tuning beyond the controlled simulation benchmark.

4.4 Ablations on LIBERO

The main LIBERO comparison shows that PhaseLoRA improves over static LoRA, including a matched-parameter High-rank LoRA baseline. We further ablate the design choices of PhaseLoRA on LIBERO-Spatial. These ablations test whether the improvement requires meaningful descriptor alignment, whether weak proxy supervision is useful, whether the two descriptors and their interaction contribute, and whether descriptor-dependent scalar rescaling is sufficient.

Descriptor randomization. We first test whether the descriptors must be aligned with the underlying control regime. In the random-descriptor control, the descriptor values used by the PhaseLoRA layers are replaced with pseudo-random values sampled from $\mathrm{Uniform}(0, 1)$ for both P and E. This preserves time-varying scalar inputs and the PhaseLoRA parameterization, but removes the relationship between the descriptors and the current rollout. If the full model outperforms this control, the gain cannot be explained merely by additional parameters or arbitrary temporal modulation.

Scalar-gating control. We next test whether descriptor-dependent rescaling of a fixed LoRA update is sufficient. This baseline uses the same router and predicted descriptors $(\hat{P}_t, \hat{E}_t)$ as PhaseLoRA, but constrains the update to be a scalar multiple of a fixed low-rank direction:

$$\Delta W_t = g_t BA, \qquad g_t = \exp(\alpha_P \hat{P}_t + \alpha_E \hat{E}_t + \alpha_{PE} \hat{P}_t \hat{E}_t), \tag{13}$$

where A and B are standard LoRA factors, and $\alpha_P, \alpha_E, \alpha_{PE}$ are learned scalar coefficients for each adapted layer. The scalar coefficients are initialized to zero, so the baseline starts from ordinary LoRA with $g_t = 1$.

This control preserves descriptor-dependent temporal modulation, but prevents the descriptors from changing the low-rank update direction. In contrast, PhaseLoRA uses $B_t = B_0 + \hat{P}_t B_P + \hat{E}_t B_E + \hat{P}_t \hat{E}_t B_{PE}$, which changes the effective left factor. A gap between the scalar-gated baseline and PhaseLoRA therefore tests whether direction-changing adaptation is needed beyond scalar modulation.

Results. Table 3 summarizes the ablations. Random descriptors substantially underperform PhaseLoRA, indicating that arbitrary time-varying inputs are not sufficient; the descriptors must remain aligned with rollout-dependent control variation. The scalar-gated variant also underperforms PhaseLoRA, showing that descriptor-dependent rescaling of a fixed LoRA direction does not recover the benefit of changing the effective update direction.

Removing proxy supervision reduces performance, suggesting that the weak targets provide a useful training signal for the router. Using only P or only E also underperforms the full model, indicating that fine-control tendency and event/boundary intensity provide complementary information. Removing the interaction component B_{PE} yields a smaller drop, suggesting that most of the gain comes from additive descriptor conditioning, with the interaction term providing an additional refinement.

Overall, these ablations support the main design of PhaseLoRA: the improvement is not explained by matched parameter count, arbitrary temporal modulation, or scalar rescaling alone. Instead, performance depends on descriptor-conditioned changes to the effective low-rank update.

Table 3: Ablations on LIBERO-Spatial. We report suite-level success rate. Random descriptors test whether arbitrary temporal modulation is sufficient, while scalar-gated updates test whether descriptor-dependent rescaling of a fixed LoRA direction recovers the gain.

Method	Success (%)	Drop vs. Ours
High-rank LoRA	71.0 ± 1.0	14.3
Ours w/ random descriptors	66.3 ± 3.2	19.0
Ours w/ scalar-gated update	73.7 ± 1.5	11.6
Ours w/o proxy supervision	78.7 ± 3.1	6.6
Ours (P only)	80.7 ± 1.5	4.6
Ours (E only)	81.0 ± 2.0	4.3
Ours w/o PE	82.3 ± 0.6	3.0
Ours	$\mathbf{85.3 \pm 1.2}$	0.0

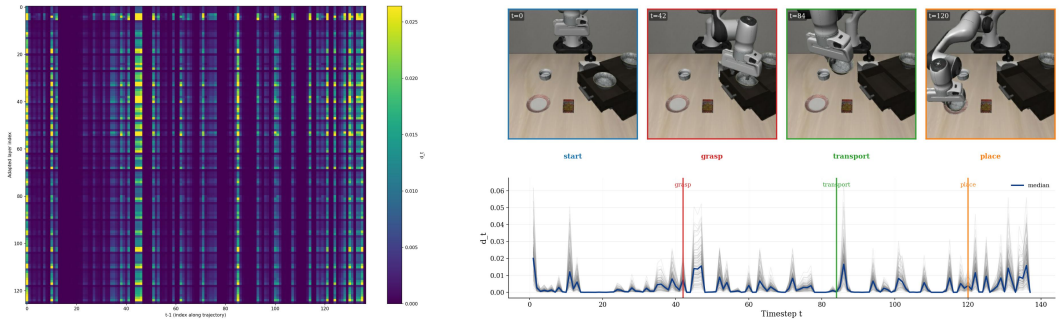


Figure 3: **Update-direction variation in PhaseLoRA on LIBERO-Spatial.** For one representative rollout, the left panel shows the adjacent-step update-direction distance d_t across adapted layers and timesteps. The upper-right row shows four rollout frames, and the lower-right panel overlays the per-layer distances with their across-layer median. Vertical lines indicate the displayed grasp, transport, and placement events. Large d_t peaks are sparse and tend to occur near these transitions.

4.5 Analysis of PhaseLoRA Updates

Update-direction variation. The scalar-gating control in Sec. 4.4 tests whether descriptor-dependent rescaling is sufficient at the level of task performance. Analytically, scalar-gated LoRA cannot change the update direction: if $\Delta W_t = g_t BA$ and $g_t > 0$, then adjacent updates remain in the same Frobenius direction. We therefore analyze whether PhaseLoRA produces structured update-direction changes over time.

For an adapted layer, let $\Delta W_t = B_t A$ denote the effective low-rank update at control step t. We measure the change in update direction between adjacent steps by the Frobenius cosine distance

$$d_t = 1 - \frac{\langle \Delta W_t, \Delta W_{t-1} \rangle_F}{\|\Delta W_t\|_F \|\Delta W_{t-1}\|_F}. \tag{14}$$

Figure 3 visualizes d_t across adapted layers and timesteps for a representative rollout from LIBERO-Spatial. Direction changes are sparse over time and occur coherently across multiple adapted layers. We observe the same qualitative pattern in the other LIBERO suites; their corresponding visualizations are provided in Appendix C. Across suites, peaks in d_t tend to appear near salient manipulation events, such as grasp, transport transition, and placement, consistent with the role of descriptor-conditioned adaptation.

Quantitative alignment. Across evaluation episodes from the four LIBERO suites, we aggregate per-layer direction distances into a timestep-level update-variation score and correlate it with descriptor changes, $|\Delta \hat{P}_t|$ and $|\Delta \hat{E}_t|$. Table 4 first verifies that the learned descriptor-dependent components are actively used: update-direction changes correlate with changes in the predicted descriptors, especially the event/boundary descriptor. More importantly, high-d_t timesteps are enriched near gripper-transition proxies for grasping and placement, suggesting that the induced direction changes are aligned with salient control events rather than being uniform temporal noise.

Table 4: Quantitative update-direction analysis across the four LIBERO suites. For each suite, statistics are computed over 100 evaluation episodes. We report Spearman correlations between update-direction changes and descriptor changes over all valid timestep samples. High-d_t timesteps are defined as the top 10% of timesteps within each suite, and transition neighborhoods are defined as ± 5 timesteps around automatically detected gripper-command transitions.

Analysis	Statistic	Spatial	Object	Goal	10
Descriptor alignment	Spearman $\rho(d_t, \lvert\Delta \hat{P}_t\rvert)$	0.502	0.439	0.497	0.440
	Spearman $\rho(d_t, \lvert\Delta \hat{E}_t\rvert)$	0.712	0.743	0.709	0.737
Transition enrichment	High-d_t near transitions	21.0%	24.3%	21.8%	24.3%
	High-d_t outside transitions	8.1%	7.9%	8.7%	9.0%
	Enrichment	2.60×	3.10×	2.52×	2.71×

5 Conclusion

We introduced PhaseLoRA, a parameter-efficient fine-tuning method for continuous-action VLA policies that conditions LoRA updates on weakly supervised descriptors of fine-control tendency and event/boundary intensity. By modulating adaptation with trajectory-level control heterogeneity, PhaseLoRA improves over standard LoRA, matched-parameter High-rank LoRA, and structured LoRA variants on LIBERO. Ablations and update-direction analyses support the role of descriptor-conditioned update directions, and real-world tabletop experiments suggest that the trend can transfer to physical execution.

Limitations and scope. PhaseLoRA uses action-derived weak proxy descriptors rather than manual phase labels or additional force/tactile sensing, which keeps the method lightweight but may require alternative descriptors for task families where action statistics do not reflect fine-control or boundary events. Our main comparisons cover the four LIBERO suites, while detailed ablations are mainly on LIBERO-Spatial and real-world results are limited to four tabletop tasks on one robot platform.

References

[1] 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, 27–30 Sep 2025.

[2] 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 J. 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 T. Tran, Vincent Vanhoucke, Steve Vega, Quan 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 Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems*, 2023. ISBN 978-0-9923747-9-2.

[3] Eric L. Buehler and Markus J. Buehler. X-LoRA: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *APL Machine Learning*, 2(2):026119, 05 2024. ISSN 2770-9019. doi: 10.1063/5.0203126.

[4] Remi Cadene, Simon Aliberts, Francesco Capuano, et al. LeRobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026.

[5] Tailai Cheng, Kejia Chen, Lingyun Chen, Liding Zhang, Yue Zhang, Yao Ling, Mahdi Hamad, Zhenshan Bing, Fan Wu, Karan Sharma, and Alois Knoll. TacUMI: A multi-modal universal manipulation interface for contact-rich tasks, 2026.

[6] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-e: An embodied multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 23–29 Jul 2023.

[7] Konstantin Gubernatorov, Mikhail Sannikov, Ilya Mikhalchuk, Egor Kuznetsov, Makar Artemov, Ogunwoye Faith Ouwatobi, Marcelino Fernando, Artem Asanov, Ziang Guo, and Dzmitry Tsetserukou. HapticVLA: Contact-rich manipulation via vision-language-action model without inference-time tactile sensing, 2026.

[8] Peng Hao, Chaofan Zhang, Dingzhe Li, Xiaoge Cao, Xiaoshuai Hao, Shaowei Cui, and Shuo Wang. TLA: tactile-language-action model for contact-rich manipulation. *Robot Learning*, 3 (1), 2026. doi: 10.55092/rl20260001.

[9] Bing Hu, Zaijing Li, Rui Shao, Junda Chen, April Hua Liu, Wei-Shi Zheng, and Liqiang Nie. From abstraction to instantiation: Learning behavioral representation for Vision-Language-Action model, 2026. URL `https://arxiv.org/abs/2605.22671`.

[10] 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*. OpenReview.net, 2022.

[11] Jiaheng Hu, Jay Shim, Chen Tang, Yoonchang Sung, Bo Liu, Peter Stone, and Roberto Martin-Martin. Simple recipe works: Vision-language-action models are natural continual learners with reinforcement learning. *arXiv preprint arXiv:2603.11653*, 2026.

[12] Dongchi Huang, Zhirui Fang, Tianle Zhang, Yihang Li, Lin Zhao, and Chunhe Xia. Co-rft: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning. *arXiv preprint arXiv:2508.02219*, 2025.

[13] Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-VLA: Unlocking vision-language-action model's physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025. doi: 10.48550/arXiv.2507.09160.

[14] Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew Peters. HINT: Hypernetwork instruction tuning for efficient zero- and few-shot generalisation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11272–11288, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.631.

[15] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 991–1002. PMLR, 08–11 Nov 2022.

[16] Xiaolong Jin, Kai Wang, Dongwen Tang, Wangbo Zhao, Yukun Zhou, Junshu Tang, and Yang You. Conditional lora parameter generation. *arXiv preprint arXiv:2408.01415*, 2024.

[17] Donghoon Kim, Minji Bae, Unghui Nam, Gyeonghun Kim, Suyun Lee, Kyuhong Shim, and Byonghyo Shim. Adaptive capacity allocation for vision language action fine-tuning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

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

[19] 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, 06–09 Nov 2025.

[20] Hao Li, Ziqin Wang, Zi-han Ding, Shuai Yang, Yilun Chen, Yang Tian, Xiaolin Hu, Tai Wang, Dahua Lin, Feng Zhao, Si Liu, and Jiangmiao Pang. Robointer: A holistic intermediate representation suite towards robotic manipulation. In *International Conference on Learning Representations*, 2026.

[21] Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. VLA-RFT: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators, 2025.

[22] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: benchmarking knowledge transfer for lifelong robot learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[23] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32100–32121. PMLR, 21–27 Jul 2024.

[24] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a diffusion foundation model for bimanual manipulation. *ArXiv*, abs/2410.07864, 2024.

[25] Chuancheng Lv, Lei Li, Shitou Zhang, Gang Chen, Fanchao Qi, Ningyu Zhang, and Hai-Tao Zheng. HyperLoRA: Efficient cross-task generalization via constrained low-rank adapters generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16376–16393, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.956.

[26] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, 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. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[27] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024.

[28] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 785–799. PMLR, 14–18 Dec 2023.

[29] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.239.

[30] Jan Ole von Hartz, Tim Welschehold, Abhinav Valada, and Joschka Boedecker. The art of imitation: Learning long-horizon manipulation tasks from few demonstrations. *IEEE Robotics and Automation Letters*, 9(12):11369–11376, 2024. doi: 10.1109/LRA.2024.3487506.

[31] Yongchen Wang, Kangyi Lu, Lan Wei, and Dandan Zhang. Mag-VLA: Vision-Language-Action model for bimanual magnetically actuated microrobot manipulation, 2026. URL `https://arxiv.org/abs/2605.28486`.

[32] Yake Wei, Yu Miao, Dongzhan Zhou, and Di Hu. Moka: Multimodal low-rank adaptation for mllms. In *Advances in Neural Information Processing Systems*, 2025.

[33] Xun Wu, Shaohan Huang, and Furu Wei. Mixture of LoRA experts. In *The Twelfth International Conference on Learning Representations*, 2024.

[34] Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, Wenqiang Zhang, and Cewu Lu. ForceVLA: Enhancing VLA models with a force-aware MoE for contact-rich manipulation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[35] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023.

[36] Zifan Zhao, Siddhant Haldar, Jinda Cui, Lerrel Pinto, and Raunaq M. Bhirangi. Touch begins where vision ends: Generalizable policies for contact-rich manipulation. *arXiv preprint arXiv:2506.13762*, 2025.

[37] 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, 06–09 Nov 2023.

A Implementation and Reproducibility Details

This section provides additional implementation details for reproducing the LIBERO experiments.

A.1 Backbone and action representation

All experiments are built on the $\pi_{0.5}$ VLA backbone. We initialize the model from the public base checkpoint at `gs://openpi-assets/checkpoints/pi05_base/params`. The VLM component uses the `gemma_2b_lora` variant, and the action expert uses the `gemma_300m_lora` variant.

Images are resized to 224×224 during both training and evaluation. For LIBERO evaluation, the policy receives two real RGB camera views, `base_0_rgb` and `left_wrist_0_rgb`. The third image slot, `right_wrist_0_rgb`, is filled with zeros as a dummy view for compatibility with the model input format. Language instructions are taken directly from the LIBERO task language annotations.

The model predicts action chunks with action dimension 32 and horizon $H = 10$. For LIBERO control, we use the first 7 action dimensions, corresponding to 3 translational dimensions, 3 rotational dimensions, and 1 gripper command. PhaseLoRA itself does not require a particular rotation parameterization. In our LIBERO implementation, however, the first seven action dimensions are

$$\mathbf{a}_t = [\Delta x_t,\, \Delta y_t,\, \Delta z_t,\, \Delta r_t^x,\, \Delta r_t^y,\, \Delta r_t^z,\, g_t] \in [-1, 1]^7.$$

The first six dimensions follow the normalized delta-command convention of the robosuite v1.4.1 `OSC_POSE` controller. The first three components specify relative Cartesian-position commands and are scaled by the controller to a maximum absolute value of $0.05\,\mathrm{m}$ per dimension. The next three components specify a relative axis–angle rotation vector and are scaled to a maximum absolute value of $0.5\,\mathrm{rad}$ per dimension. The gripper command g_t lies in $[-1, 1]$, where -1 denotes opening and $+1$ denotes closing. LIBERO actions are already represented as delta commands, so we do not apply an additional delta-action transformation. During training, actions are normalized per dimension using training-set quantile statistics. At inference time, the predicted actions are inverse-normalized before their first seven dimensions are passed to the environment. During closed-loop evaluation, the policy predicts a 10-step action chunk but executes only the first 5 low-level actions before replanning. Thus, the default replanning interval is 5 control steps.

A.2 LIBERO benchmark and evaluation protocol

We evaluate on the four standard LIBERO suites used in the main paper: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10. We use the official LIBERO datasets and task language annotations from the LIBERO benchmark. Each suite-specific policy is fine-tuned on the corresponding LIBERO training split and evaluated on the corresponding suite.

Unless otherwise stated, evaluation uses 10 trials per task and 3 seeds. The success rate for a suite is computed as the episode-level success rate,

$$\mathrm{SuccessRate} = \frac{\text{\# successful evaluation episodes}}{\text{\# total evaluation episodes}}.$$

The average reported in the main table is the average across the four LIBERO suite scores.

A.3 LoRA and PhaseLoRA configuration

Our implementation follows the LoRA implementation in the Gemma modules of the $\pi_{0.5}$ codebase. Instead of specifying a separate PEFT-style list of target modules, the attention and feed-forward linear projections in the Gemma implementation are wrapped with LoRA modules. Specifically, LoRA is applied to the attention projections, including the query/key/value and output projections, as well as to the feed-forward gate and linear projections. We do not attach a separate LoRA module to an additional action head.

For standard LoRA and PhaseLoRA, the VLM LoRA rank is 48 with scaling factor $\alpha = 48$, and the action-expert LoRA rank is 96 with scaling factor $\alpha = 96$. For the matched-parameter High-rank LoRA baseline, the VLM LoRA rank is 80 with $\alpha = 80$, and the action-expert LoRA rank is 160 with $\alpha = 160$. The reported trainable-parameter percentages are measured relative to the full $\pi_{0.5}$ backbone parameter count. The LoRA implementation does not use LoRA dropout.

For PhaseLoRA, each adapted layer uses four left factors and one shared right factor. Given router-predicted descriptors $(\hat{P}_t, \hat{E}_t)$ at control step t, the effective left factor is

$$B_t = B_0 + \hat{P}_t B_P + \hat{E}_t B_E + \hat{P}_t \hat{E}_t B_{PE},$$

and the low-rank update is

$$\Delta W_t = B_t A.$$

Thus, for a fixed pair $(\hat{P}_t, \hat{E}_t)$, the update remains low-rank, while its effective left factor can vary across control steps. The shared LoRA component follows the standard LoRA initialization: the right factor is initialized with small Gaussian noise and the corresponding left factor is initialized to zero. The descriptor-dependent left factors B_P, B_E, and B_{PE} are initialized to zero, so the model starts from an ordinary LoRA update and learns descriptor-dependent deviations only during fine-tuning.

A.4 Router architecture

The phase router predicts two scalar descriptors at each control step: a fine-control descriptor $\hat{P}_t$ and an event/boundary descriptor $\hat{E}_t$. The router uses three input branches: a visual-language prefix summary, recent action and gripper history. The visual-language summary is computed by masked mean pooling over the multimodal prefix embeddings. For the `gemma_2b_lora` VLM used in our

experiments, this prefix summary has dimension 2048. The recent-action branch uses the previous 6 action chunks, with the first 5 executed actions retained from each chunk, yielding a history window of 30 low-level action steps.

For each low-level action step in the history window, the router forms a 23-dimensional feature: 7 action dimensions, 7 first-order differences, 7 second-order differences, 1 gripper state feature, and 1 gripper-change feature. The prefix summary and action-history features are each projected to the router hidden dimension of 128. The temporal action-history branch is then processed by a single-layer, unidirectional GRU with hidden dimension 128. The fused router hidden state is passed to two linear heads, followed by sigmoid activations, to produce $\hat{P}_t, \hat{E}_t \in [0, 1]$.

At the beginning of an episode, when fewer than 30 previous low-level actions are available, the history window is left-padded with zeros. If no history is available, the router receives an all-zero history window. During training, the router uses offline action history to avoid leakage from the current target action chunk. During evaluation, the action history is updated online from the actions previously executed by the policy.

A.5 Weak descriptor targets

The weak descriptor targets are precomputed offline and injected during training. The fine-control target is derived from a robustly normalized motion statistic over the trajectory. In the implementation, the translational action magnitude is combined with the rotational action magnitude using rotation weight 0.5 and with the gripper-command change using gripper weight 0.1. The event/boundary target is derived from action jerk.

All trajectory statistics are robustly normalized with median and median absolute deviation, using $\epsilon = 10^{-6}$ for numerical stability. The sigmoid inputs are clipped to $[-60, 60]$ before applying the sigmoid, and the resulting targets are clamped to $[0, 1]$. The gripper command is treated as a continuous command rather than a binary open/close label.

A.6 Training objective and optimization

The main policy objective follows the $\pi_{0.5}$ training objective. The policy is trained with a flow-matching mean-squared error objective, where the target velocity is defined as the difference between sampled noise and the action target. The loss is applied to the predicted action tensor as a whole; we do not use separate manually tuned weights for translation, rotation, and gripper dimensions.

The router is supervised with the weak descriptor targets using

$$\mathcal{L}_{\text{router}} = 0.5\,\text{Huber}(\hat{P}_t, P_t) + 0.2\,\text{BCE}(\hat{E}_t, E_t),$$

where the Huber delta for the fine-control target is 0.1.

All models are trained with AdamW. We use effectively no weight decay, setting it to 10^{-10}, and apply gradient clipping with maximum norm 1.0. For the descriptor-dependent LoRA regularizer in Eq. (12), we set

$$\lambda_P^{\text{reg}} = \lambda_E^{\text{reg}} = \lambda_{PE}^{\text{reg}} = 10^{-5}$$

for all LIBERO and real-world experiments. The learning rate is linearly warmed up for 10,000 steps to 5×10^{-5} and then kept constant for the remainder of training. We use batch size 32 and train each run for 30,000 optimization steps. Training uses bfloat16 precision. Exponential moving average is disabled. The training configuration resizes images to 224×224; in the PyTorch training path, non-wrist images additionally use a small random crop followed by resizing back to the target resolution. We do not use checkpoint averaging or early stopping.

A.7 Baseline fairness

For LoRA-based methods, we keep the backbone, data split, image preprocessing, action representation, training objective, batch size, learning-rate schedule, number of training steps, and adapted module family fixed. Standard LoRA and PhaseLoRA use the same base LoRA ranks, while High-rank LoRA increases the ranks to approximately match the trainable-parameter budget of PhaseLoRA. LoRA-SP is applied to the same family of Gemma attention and feed-forward modules. Its key settings are energy threshold 0.9, router hidden dimension 256, spectral loss weight 10^{-2}, router loss weight 10^{-3}, and inference-time pruning enabled.

A.8 Compute resources

All LIBERO experiments were run on NVIDIA RTX 5090 GPUs. Each training run used a single GPU with bfloat16 precision and batch size 32. The peak GPU memory usage of PhaseLoRA was approximately $22106\,\mathrm{MiB}$ per run. A single suite-specific LIBERO training run of 30,000 optimization steps took approximately 24 GPU-hours, depending on the method and suite.

For the PhaseLoRA main result, one seed across the four LIBERO suites required approximately $4 \times 24 = 96$ GPU-hours, and the three-seed PhaseLoRA result required approximately 288 GPU-hours. The full main LIBERO comparison in Table 1 includes six methods, four suites, and three seeds, requiring approximately

$$6 \times 4 \times 3 \times 24 = 1728$$

GPU-hours. The additional LIBERO-Spatial ablations in Table 3 required approximately

$$6 \times 1 \times 3 \times 24 = 432$$

GPU-hours. Thus, the reported LIBERO training runs required approximately 2160 GPU-hours in total.

The experiments used approximately 100GB of local SSD storage for the LIBERO datasets, the $\pi 0.5$ pretrained checkpoint, training checkpoints, logs, and cached artifacts. Real-world evaluation was run on the robot platform and did not require additional large-scale model training beyond the fine-tuned checkpoints. Preliminary debugging runs and failed experiments required additional compute, but they are not included in the totals above because they were not part of the final reported experimental protocol.

B Real-world Benchmark Details

This section provides additional details for the real-world manipulation benchmark used in Sec. 4.3. The real-world experiments are intended as supporting evidence for the controlled LIBERO results, rather than as a claim of broad real-world generalization across robot platforms or task families.

Robot platform. We use an AgileX Piper robotic arm equipped with a two-finger gripper. The robot is mounted on a tabletop workspace and evaluated in closed-loop execution. All real-world policies are initialized from the same pretrained VLA backbone as the LIBERO experiments and are fine-tuned using the same method-specific adapter configuration as in the simulation benchmark, unless otherwise stated.

Observation setup. The policy receives RGB observations from two cameras: a wrist-mounted RGB camera near the gripper and a fixed external RGB camera observing the tabletop workspace. Example views of the hardware and workspace are shown in Fig. 4. All RGB images are resized to 224×224 before being passed to the VLA policy. The policy uses only RGB observations and the language instruction. We do not provide proprioceptive state, force sensing, or tactile sensing to the policy during inference.

Tasks. The benchmark contains four tabletop manipulation tasks: moving a tennis ball from a yellow plate to a blue plate, removing a cuboid from a blue plate, picking a red cube into a yellow plate, and putting an apple on a yellow plate. These tasks require grasping, transport, object release, and placement under real robot execution. Each task contains 100 demonstrations for fine-tuning. Demonstrations are collected using the same camera setup and robot control interface used during policy evaluation.

Action chunking and closed-loop control. For real-world evaluation, the policy predicts action chunks with horizon $H = 50$. During closed-loop execution, we execute the first 10 low-level actions from each predicted chunk and then replan. Thus, each policy query produces a 50-step action chunk, while the robot is controlled in a receding horizon manner with a replanning interval of 10 executed low-level control steps. This chunking setup is kept fixed across all compared methods in the real-world benchmark.



(a) Piper arm with wrist camera.

(b) Full tabletop setup.

(c) External-camera view.

Figure 4: Real-world benchmark setup. (a) Piper robotic arm with a two-finger gripper and a wrist-mounted RGB camera. (b) Full tabletop setup with a fixed external RGB camera, lighting, task objects, and the robot workspace. (c) Example external-camera observation showing the plates, objects, and distractors used in evaluation. The policy uses two RGB views and language instructions, without proprioceptive, force, or tactile inputs.

Training and baseline fairness. For each method, the policy is fine-tuned on the same set of demonstrations for the corresponding task. The compared methods use the same pretrained backbone, observation inputs, language instructions, training data, action chunking setting, optimizer, batch size, learning-rate schedule, and number of training steps. The High-rank LoRA baseline is chosen to approximately match the trainable-parameter budget of PhaseLoRA, so the real-world comparison tests whether the observed trend is explained solely by increased adaptation capacity.

Evaluation protocol. For each method and task, we evaluate 10 physical trials for each of 3 random seeds, resulting in 30 trials per task per method. Success is judged manually according to pre-defined task-specific criteria. An episode is counted as successful if the target object is moved to or removed from the specified target region by the end of the episode. Episodes are counted as failures if the robot misses the grasp, drops the object, places the object in the wrong region, causes an unrecoverable collision, or reaches the time limit. Manual judgments are based on direct observation and recorded evaluation videos. No evaluation episodes are used for training or hyperparameter selection.

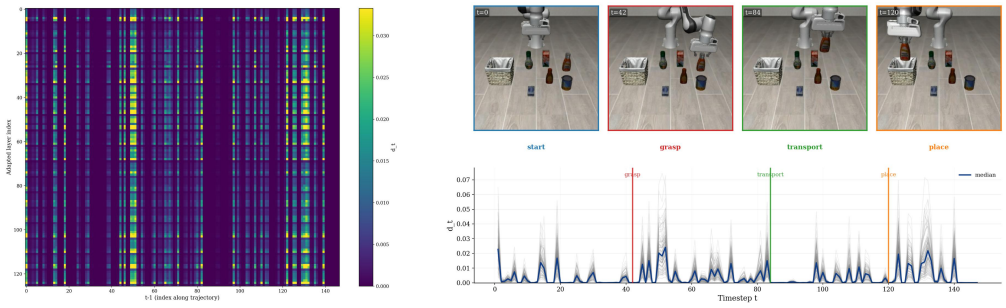
Limitations of the real-world benchmark. The real-world benchmark is limited to four tabletop tasks on a single robot platform and a single workspace setup. It is therefore used to test whether the matched-capacity trend observed in LIBERO also appears under physical execution, rather than to establish general real-world robustness across robots, camera placements, object categories, or environments.

C Additional Update-Direction Visualizations

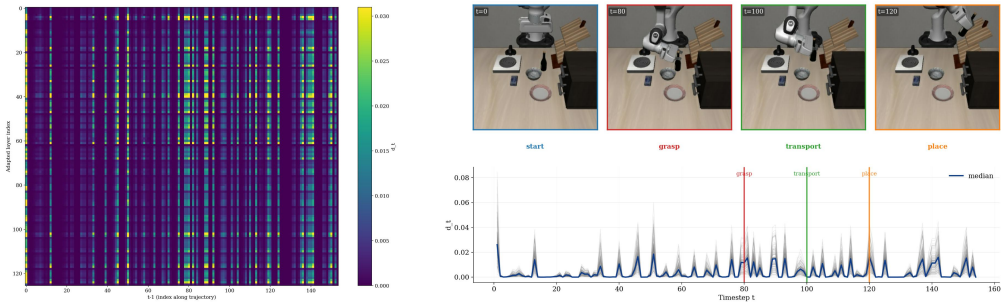
Figure 5 provides additional update-direction visualizations for LIBERO-Object, LIBERO-Goal, and LIBERO-10. Consistent with the LIBERO-Spatial example in the main text, update-direction changes are sparse over time, coherent across multiple adapted layers, and tend to occur near salient manipulation events such as grasping, transport transitions, and placement.

D Broader impacts

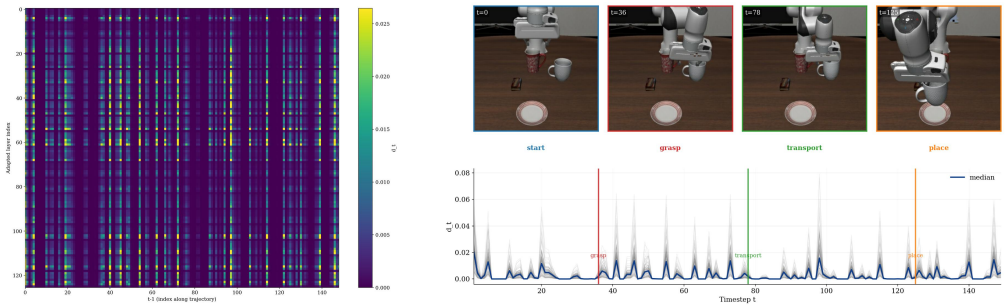
PhaseLoRA aims to make downstream adaptation of continuous-action VLA policies more parameter-efficient and effective. Potential positive impacts include reducing the computational cost of adapting robot policies and making manipulation systems more practical for research and applications such as assistive robotics, laboratory automation, and manufacturing. At the same time, improving robot policy adaptation may lower the barrier to deploying robots in physical environments, where failures can cause material damage or safety risks if systems are used outside their validated operating



(a) LIBERO-Object



(b) LIBERO-Goal



(c) LIBERO-10

Figure 5: **Additional update-direction visualizations for the remaining LIBERO suites.** Each row shows one representative rollout from LIBERO-Object, LIBERO-Goal, or LIBERO-10. In each row, the left panel shows the adjacent-step update-direction distance d_t across adapted layers and timesteps, the upper-right row shows four rollout frames, and the lower-right panel overlays the per-layer distances with their across-layer median. Vertical lines indicate the displayed grasp, transport, and placement events. As in the main-text LIBERO-Spatial example, large d_t peaks are sparse and tend to occur near these transitions.

conditions. The method should therefore be evaluated with task-specific safety checks, human oversight, and deployment constraints before use in real-world settings. We do not use personally identifiable data or human-subject data in the reported experiments, but broader deployment of robotic systems may raise labor, safety, and accountability concerns that depend on the application context.

E Existing assets and licenses

We use several existing assets and follow their corresponding licenses and terms of use. The LIBERO benchmark is used for the main controlled experiments; we cite the original LIBERO paper and use the official LIBERO datasets and task annotations. According to the official LIBERO resources, the LIBERO codebase is released under the MIT License and the datasets are released under Creative

Commons Attribution 4.0 International (CC BY 4.0). We use the standard LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10 suites.

Our experiments build on the public OpenPI implementation and the public $\pi 0.5$ base checkpoint. We cite the original $\pi 0.5$ work and initialize from the public base checkpoint described in Appendix A.1. OpenPI is released under the Apache License 2.0, and the repository also includes Gemma license and terms files for Gemma-based components. We do not redistribute the original pretrained checkpoint or LIBERO datasets in our supplementary code release. Instead, the supplementary material provides instructions for obtaining these assets from their original sources and for using them in accordance with their respective licenses and terms of use.