PRISM: Primitive Routing via In-context Skill Mixing for Retentive VLA Adaptation

Jiachen Tao **Gengyu Zhang** **Haoxuan Wang**
Bing Liu **Yan Yan**[†]
University of Illinois Chicago

Abstract

Vision-Language-Action (VLA) models promise a single language-conditioned policy adaptable to many manipulation behaviors, yet deployment additionally requires *skill retention* and *long-horizon composition*: a policy must retain previously acquired skills and reuse them inside composite tasks whose behavior changes across phases. The standard recipe absorbs every new task into one shared update, which provides neither a parameter-level locus for reusable skills nor a mechanism for invoking each skill along the trajectory. We observe that this missing structure is latent rather than absent: per-skill low-rank updates of a pretrained VLA occupy a robustly distinguishable, low-dimensional subspace, and a single composite-task LoRA concentrates roughly half of its Frobenius energy in this subspace, against a near-zero matched random-rotation control. Building on this geometry, we introduce **PRISM**, which represents reusable skills as parameter-level primitives: each skill family is stored as a dedicated low-rank action-expert adapter in a *primitive bank*, primitives are augmented with skill-specific low-rank residuals and composed at inference by a per-chunk *VLM-conditioned router* that reads the prefix representation to track task progress, and the router is supervised by per-chunk labels from a deterministic, task-conditioned, action-derived oracle. During composite training, the vision encoder, the language LoRA, the skill residuals, the router, and the shared action projections and time MLP are updated; every primitive is held frozen, so the base per-skill action-expert adapters are preserved by construction. On RoboCasa365, PRISM achieves $\mathbf{56.43}\%$ atomic-skill success and reaches $\mathbf{20.00}\%$ on long-horizon composite tasks, while a standard $\pi_{0.5}$-LoRA two-stage fine-tune reaches 17.86% atomic and 17.50% composite success. We additionally validate PRISM on an AgileX PIPER real-robot setup.

1 Introduction

Vision-language-action (VLA) models [3–6, 20, 35, 36] are emerging as a general substrate for language-conditioned robot manipulation: a single pretrained policy, steered by natural-language instruction, can in principle be adapted toward a broad range of downstream behaviors. Yet deployment exposes two failure modes of the standard single-policy adaptation paradigm: sequentially fine-tuning the policy on a new behavior degrades performance on previously acquired ones, forcing a choice between retraining from scratch and losing capabilities; and long-horizon tasks that recombine known sub-skills must still be fit end-to-end as if their constituent skills had not already been mastered—an object-washing task, for instance, requires end-to-end training even though it chains a lever-like interaction (to activate a water source) with a pick-and-place-like interaction (to move the object through the stream). We refer to the two requirements that address these failures as *skill retention* and *long-horizon composition*, and ask how a VLA should be parameterized to support both.

[†]Corresponding Author

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

The standard adaptation paradigm is monolithic: a downstream task is absorbed into a shared set of model parameters, with no explicit representation of which part of the update corresponds to which reusable behavior. This parameterization may be adequate for mastering one task in isolation, but it lacks two handles required for deployment. The first is *where*: where is a previously learned skill localized in the model update? Without such a parameter-level locus, later updates can interfere with the directions that supported earlier behaviors. Common defenses either allocate additional task-specific parameters, which limits reuse, or constrain updates globally, which can slow acquisition without identifying the skill-specific directions that require protection. The second handle is *when*: when during a long-horizon trajectory should the policy invoke each skill? A composite task cycles through distinct phases of object interaction, motion, and sub-goal. Without a mechanism for switching behavioral regimes, an undifferentiated policy must fit each composite end-to-end as if it were a fresh task. Long-horizon demonstrations are not merely longer action sequences but reflect the selection and recombination of behaviors the policy has already acquired, and a policy with no parameter-level skill identity has no way to interpret such a trajectory in relation to its prior skills.

We find that this missing structure is more latent than absent. When each primitive skill family is adapted on its own through a small low-rank update [17], the family of per-skill update tensors does not diffuse uniformly throughout parameter space: it occupies a robust, semantically clustered subspace that reproduces across independently trained checkpoints (Section 3.3)—low-dimensional relative to the full action-expert parameter space, while remaining nearly full-rank across primitive families. Moreover, when a single LoRA is fit on a composite-task dataset, a substantial fraction of its update tensor lies in the linear span of the per-skill update tensors (Section 3.3): composite-task behavior is, to a first approximation, partially expressible as a combination of per-skill primitives. However, when atomic and composite behaviors are absorbed into a single shared adapter trained sequentially, the skill-specific directions learned during atomic adaptation are no longer separately addressable, and atomic-skill performance degrades as composite training proceeds. We refer to this loss of per-skill addressability as *primitive collapse*.

This observation suggests a constructive parameterization. If reusable skills occupy distinguishable low-dimensional update directions, then mixture weights over these updates can be trained to serve as skill-level control variables rather than arbitrary adapter coefficients. Later adaptation need not rewrite a shared update; it can preserve skill-localized slots while learning to route over them. Likewise, long-horizon execution need not fit composite behavior as an independent policy; it can select or mix primitives over time as the task progresses. Under this view, retaining and composing skills reduce to *routing over preserved primitives*: once a skill is represented as a primitive slot, executing a long-horizon behavior activates the appropriate primitive or mixture at each phase.

Building on this observation, we introduce **PRISM**, a VLA adaptation framework that turns reusable skills into first-class parameter-level objects. PRISM has three coupled components: a *primitive bank* that stores each skill family as its own low-rank adapter over the action expert, augmented with a skill-specific residual during composite training; a *prefix-conditioned router* that emits a per-chunk mixture over these primitive–residual branches; and a *deterministic action-derived labeler* that supplies per-chunk supervision for the router. The bank gives a direct parameter-level answer to the *where* question—adding a new skill amounts to adding a new slot, while every existing primitive remains parametrically untouched.

The router answers the *when* question. At every action chunk it reads the VLM prefix—a contextual feature fusing the current instruction with the current visual state—and emits a distribution over the bank, so the active mixture varies along a long-horizon trajectory. Conditioning on the prefix rather than on the instruction is essential: the instruction is fixed across the trajectory, while sub-goal placement depends on how far the robot has progressed. Training the router from composite demonstrations is nontrivial because behavior cloning alone exerts little pressure for mixture weights to align with skill semantics; PRISM therefore supervises it with per-chunk labels produced by a deterministic, action-derived labeler that requires no learned parameters or per-trajectory human labels once the rule is fixed.

In summary, PRISM makes three contributions. **(i) Structural finding.** Per-skill low-rank updates of VLA action experts span a near-N-dimensional, semantically clustered subspace that is reproducible across independently trained banks; a composite-task LoRA places $\sim 56\%$ of its Frobenius energy in this span, versus $\approx 0\%$ under matched random rotations. **(ii) Framework.** PRISM couples a primitive bank with skill-specific residuals, a per-chunk VLM-conditioned router, and a deterministic

action-derived labeler; during composite training, the vision encoder, language LoRA, residuals, router, action projections, and time MLP are updated, while primitives remain frozen. **(iii) Evaluation.** On RoboCasa365, PRISM achieves $\mathbf{56.43}\%$ atomic success after composite training and reaches $\mathbf{20.00}\%$ composite success, against 17.86% atomic and 17.50% composite success for a standard $\pi_{0.5}$-LoRA two-stage fine-tune. We further validate PRISM on an AgileX PIPER real robot.

2 Related Work

Skill Libraries and Skill Composition. PRISM is closest to methods that represent skills as explicit reusable modules. AtomicVLA [53] routes via skill descriptor tokens; SkillVLA [52] decouples bimanual action experts; PSEC [30] composes diffusion-policy LoRA modules in parameter space. PRISM differs in three coupled choices: each skill family is stored as an action-expert LoRA primitive augmented with a trainable residual during composite adaptation; routing is over these branches at action-chunk granularity using the evolving VLM prefix, so the active mixture changes with visual progress rather than with a task or skill identifier; and adapted attention contributions and complete skill-specific FFN outputs are blended in activation space, avoiding cross-products between LoRA factors from different branches. Earlier skill-library and segmentation methods discover, segment, or reuse skills from demonstrations [32, 33, 44, 45]; PRISM instead assumes a predefined skill taxonomy and tests whether such families are parametrically separable in a pretrained VLA. Concurrent PEFT-library methods DMPEL [22] and CLARE [41] grow and route over adapter libraries for lifelong robot learning; PRISM instead targets retention-centric long-horizon composition with chunk-level routing within trajectories rather than acquiring a sequence of discrete tasks.

Parameter-Efficient Fine-Tuning and Mixture of Experts. PRISM builds on parameter-efficient adaptation and expert routing. Low-rank adaptation [17], together with adapter-based and quantized variants [11, 16], provides a compact interface for adapting large pretrained models without full-model updates. Mixture-of-experts architectures route computation through specialized modules in language and multimodal models [10, 13, 19], and related expert-routing ideas have also been explored in robot policies [18, 40, 49]. PRISM treats each primitive–residual branch as a skill expert, but differs from standard MoE in two ways: routing is over skill-localized action-expert adapters at action-chunk granularity, and adapted attention contributions and complete skill-specific FFN outputs are blended in activation space rather than mixing LoRA factors in parameter space.

Continual Learning and Catastrophic Forgetting. Continual learning addresses catastrophic forgetting through regularization, rehearsal, and architecture-based expansion. Regularization methods penalize changes to parameters important to past tasks [21, 24, 50]; rehearsal methods replay past examples [7, 8]; and architecture-based methods allocate or freeze task-specific parameters [31, 42]. In robotics, lifelong learning benchmarks and imitation pipelines study sequential skill acquisition and retention [27, 45]. PRISM is closest in spirit to architecture-based continual learning, but targets a different VLA setting: atomic skills must be retained while being reused inside composite trajectories. Loss- and sample-level defenses such as replay can mitigate forgetting, but they do not provide a skill-localized parameter interface for chunk-level composition. PRISM instead freezes per-skill action-expert primitives during composite training, trains a residual alongside each primitive, and learns to route over these branches, preserving the primitive parameters by construction while evaluating behavioral retention empirically.

3 Preliminaries and Empirical Skill Geometry

This section establishes the empirical foundation for PRISM: we define the retention-centric VLA adaptation setting, introduce the LoRA update tensors used for analysis, and show that per-skill updates form a reproducible subspace that captures a substantial fraction of composite-task adaptation.

3.1 Problem Setup and Notation

Atomic skills and composite tasks. We assume access to N *atomic skills* $\mathcal{S} = \{s_1, \ldots, s_N\}$ (skill families at the task level; we use *primitive* for their LoRA-form parametric representation, introduced in §3.2), each paired with demonstrations D_{s_i}, and a set of *composite tasks* $\mathcal{T}$ with demonstrations $D_{\text{composite}}$ that cycle through subsets of $\mathcal{S}$ with phase boundaries unlabeled at the chunk level. The retention setting requires that, after composite training on $\mathcal{T}$, per-skill success rate on $\mathcal{S}$ does not

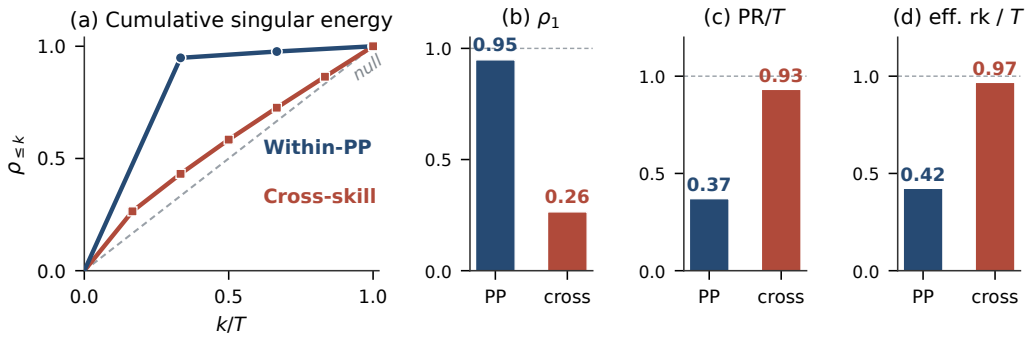


Figure 1: **Within-skill LoRAs concentrate; across-skill LoRAs spread across N. (a)** Cumulative singular energy $\rho_{\le k}$ vs k/T: within-PP saturates at the first component, while cross-skill rises along the uniform null $\rho_{\le k}{=}k/T$ (dashed). **(b)** Top-1 energy ρ_1: 0.948 vs 0.265. **(c)** PR$/T$ (1.0=maximal spread, $1/T$=collapse): 0.37 vs 0.93. **(d)** Effective rank$/T$: 0.42 vs 0.97. Across all metrics, within-skill LoRAs are more concentrated than across-skill LoRAs.

degrade; the composition setting requires that performance on $\mathcal{T}$ approaches the per-skill ceiling that ground-truth chunk-level labels would provide.

VLA backbone ($\pi_{0.5}$ substrate). We instantiate PRISM on the $\pi_{0.5}$ family [3], which couples a vision encoder ϕ_{V}, a language model ϕ_{L}, and a small action expert π_{A} that hosts the per-skill adapters in PRISM. At every chunk boundary t, the language transformer processes visual and instruction tokens to condition π_{A}, while the router reads their pooled embedding-level prefix $h_t \in \mathbb{R}^{d_{\text{prefix}}}$ before the language transformer (Equations 3–4). The action expert outputs chunks of $H = 10$ actions, of which 5 are executed before each re-plan; we use *chunk-level* for any quantity computed once per re-plan.

Adaptation interface. The pretrained language-model core and the action-expert backbone $W_{A,0}$ are frozen end to end. Adaptation touches the vision encoder ϕ_{V} (fully fine-tuned), a shared language LoRA on ϕ_{L}, action projections and time MLP, and per-skill action-expert adapters over $W_{A,0}$: primitives in Phase 1, followed by residuals and the router in Phase 2. Which subset is trainable depends on the training phase (Appendix A).

3.2 Per-Skill LoRA Adaptation

For any layer of the action expert with frozen weight $W_{A,0} \in \mathbb{R}^{d_o \times d_i}$, a rank-$r$ low-rank adaptation [17] augments the layer with a residual $\Delta W = BA$ (so that $W_A = W_{A,0} + \Delta W$), where $B \in \mathbb{R}^{d_o \times r}$ and $A \in \mathbb{R}^{r \times d_i}$. We refer to ΔW as the layer's *update tensor*. Two protocols are studied. *Per-skill adaptation* jointly trains all pairs $\{(B_i, A_i)\}$ on the union $\bigcup_i D_{s_i}$ under behavior cloning; ground-truth skill labels ensure each batch sample contributes gradients only to its own pair (B_i, A_i), so we recover the per-skill update tensor $\Delta W_i = B_i A_i$. *Composite adaptation* fits a single pair $(B_{\text{comp}}, A_{\text{comp}})$ on $D_{\text{composite}}$ at the same rank, yielding the composite update tensor ΔW_{comp}.

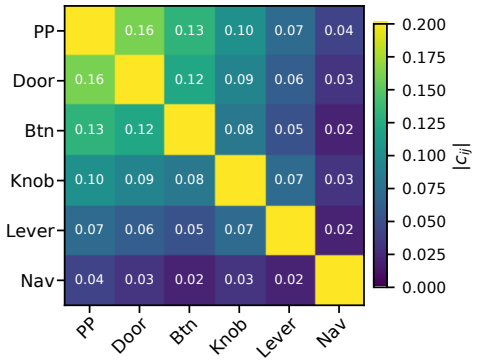


Figure 2: **Skill geometry, N=6 bank.** Pairwise cosine c_{ij} between layer-wise-flattened per-skill update tensors ΔW_i. Five manipulation skills form a coherent cluster (0.05–0.16); navigation is a clear outlier (≤ 0.043).

3.3 Empirical Skill Geometry

We measure how the per-skill update tensors $\{\Delta W_i\}_{i=1}^{N}$ occupy parameter space, and then ask how much of ΔW_{comp} lies in their linear span. Measurements come from two independently trained banks (N=6 and N=11), allowing us to test whether the resulting geometry reproduces across runs.

Within-skill task collinearity. Within a single skill family, per-task LoRAs collapse to a near-1D subspace ($\rho_1^{\text{PP}} = 0.948$, $\text{PR}_{\text{PP}} = 1.11$ on a T=3 Pick & Place pilot), in stark contrast to the cross-skill

stack ($\rho_1^{\text{cross}} = 0.265$, $\text{PR}_{\text{cross}} = 5.59 \approx N$); see Figure 1. This T=3 pilot provides initial evidence that task variation within a skill family is far more concentrated than variation across skill families; we therefore adopt skill families, rather than individual tasks, as the primitive unit in the present experiments.

Cluster geometry across independently trained checkpoints. The pairwise cosines $c_{ij} = \langle \Delta W_i, \Delta W_j \rangle_F / (\|\Delta W_i\|_F \, \|\Delta W_j\|_F)$ on the 6-skill bank reveal an interpretable manipulation cluster of five end-effector contact skills (Pick & Place, door-opening, knob-turning, button-pressing, lever-turning) with off-diagonal cosines in $[0.05, 0.16]$ and Pick & Place as the hub; navigation (robot base translation) is a clear outlier at ≤ 0.043 against every manipulation skill (Figure 2). Across the 15 skill pairs shared with the 11-skill bank, the per-pair cosine is preserved with mean $|\Delta c| = 0.016$ and max $|\Delta c| = 0.050$ (on Pick & Place $\leftrightarrow$ door-opening), indicating that the cluster structure reflects task-level motor-primitive families (end-effector contact vs. base translation) rather than a single-run optimization artefact.

Subspace dimensionality. Let $G \in \mathbb{R}^{N \times N}$ denote the primitive Gram matrix $G_{ij} = \langle \Delta W_i, \Delta W_j \rangle_F$ with eigenvalues $\{\tau_k\}_{k=1}^N$. The participation ratio

$$\text{PR} \;=\; \frac{\left(\sum_k \tau_k\right)^2}{\sum_k \tau_k^2} \;\in\; [1, N] \tag{1}$$

quantifies the number of effectively independent directions the primitives span. We measure $\text{PR} = 5.59$ at N=6 and $\text{PR} = 10.78$ at N=11 (Figure 3)—in both cases close to the maximum—so adding atomic primitives expands the spanned subspace dimension rather than redundantly populating an already-occupied direction.

Composite–primitive alignment. Let Π_{prim} denote the orthogonal projector onto the linear span of $\{\Delta W_i\}_{i=1}^N$. We report the projected energy ratio

$$\eta_{\text{comp}} \;=\; \frac{\|\Pi_{\text{prim}} \Delta W_{\text{comp}}\|_F^2}{\|\Delta W_{\text{comp}}\|_F^2}, \tag{2}$$

and a control η_{rand} obtained by replacing each ΔW_i with a random rotation that preserves Frobenius norm and singular spectrum, averaged over 8 trials per checkpoint. On both banks, $\eta_{\text{comp}} \approx 0.56$ vs $\eta_{\text{rand}} = 0.000 \pm 0.000$ (0.560 at N=6, 0.567 at N=11). The alignment is essentially N-invariant ($\Delta\eta = 0.007$ across an 83% increase in N). Composite-task structure thus lives substantially within the per-skill subspace—a parametric ground for the bank-and-router design in Section 4.

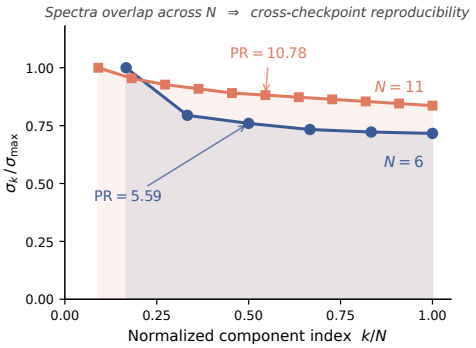


Figure 3: **Per-skill update tensors span an effectively N-dim subspace.** Normalized spectra overlap across N=6 and N=11; PR=5.59 at N=6 and PR=10.78 at N=11.

Architectural consequence. The residual $\sim$44% may reflect composite-specific structure—most plausibly transition-region dynamics that arise when the policy stitches sub-behaviors together, and that are absent from any single-skill demonstration. This residual lies outside the linear span of the frozen primitives at the parameter level, and is therefore not addressable by routing alone: any time-varying mixture $\sum_i \alpha_i(t)\, B_i A_i$ over the frozen bank still produces an effective action-expert weight within that span. The measured decomposition motivates preserving the primitive updates as a reusable foundation while providing additional trainable directions for composite adaptation. PRISM implements this separation by freezing the primitive bank and adding a skill-specific residual adapter to each primitive. The router selects mixtures of these primitive–residual branches throughout execution, while the vision encoder and language LoRA continue to adapt during composite training (Appendix A).

4 Method

Overview. At inference, PRISM takes a language instruction ℓ and camera images, and outputs action chunks of $H = 10$ actions. It operationalizes the skill geometry of Section 3 with three components:

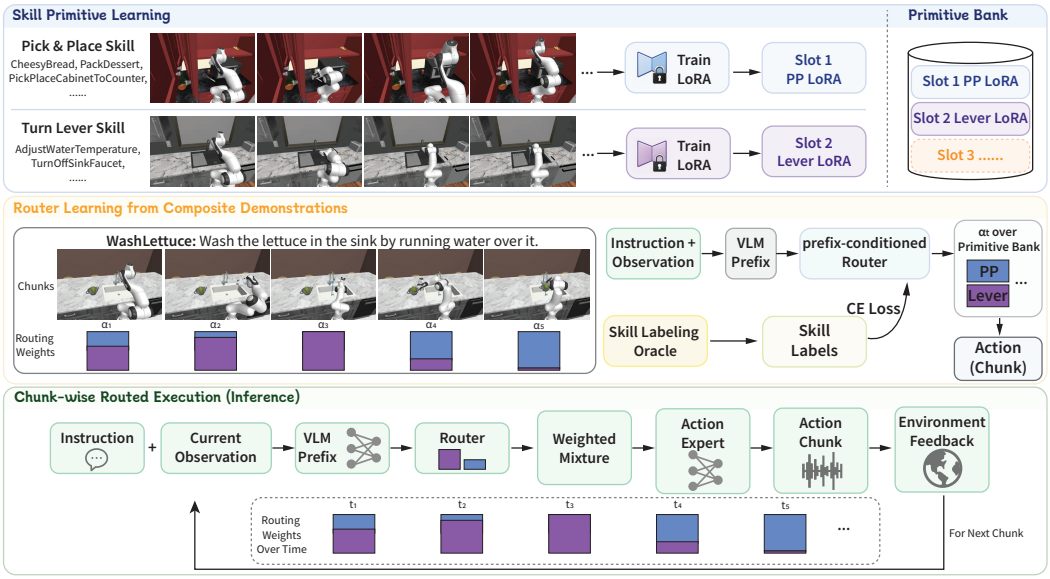


Figure 4: **Overview of the PRISM framework. (1) Skill primitive learning** (§4): each atomic skill is fitted with its own rank-r action-expert LoRA (B_i, A_i), yielding a *primitive bank* of disjoint slots, frozen after atomic training. **(2) Router learning from composite demonstrations**: a prefix-conditioned router r_θ reads the VLM prefix h_t and emits a per-chunk mixture $\alpha(t)$ over the bank, supervised by per-chunk labels from a deterministic action-derived oracle $\mathcal{A}$. **(3) Chunk-wise routed execution** (inference): each LoRA-injected action-expert layer applies the routed mixture at the activation level (Eq. 5), producing an action chunk.

a *primitive bank* that stores per-skill LoRA slots augmented with trainable residuals during composite training, a *labeling oracle* that provides skill supervision, and a *VLM-conditioned router* that selects mixtures of the primitive–residual branches. At chunk boundary t, the router reads prefix h_t and emits $\alpha(t) = r_\theta(h_t) \in \Delta^{N-1}$. These weights combine adapted attention contributions and complete skill-specific FFN outputs, as defined below. The primitives remain frozen during composite training, while their residuals provide additional adaptation capacity.

Skill Primitives and Residual Adaptation. For each atomic skill s_i, we learn a rank-32 action-expert primitive $P_i = B_i A_i$ from demonstrations D_{s_i} under one-hot routing, so each sample trains its corresponding primitive. The bank $\mathcal{B} = \{(B_i, A_i)\}_{i=1}^{N}$ consists of disjoint parameter blocks: for $i \neq j$, (B_i, A_i) and (B_j, A_j) share no parameters with each other or with the frozen action-expert backbone $W_{A,0}$. The visual encoder and shared language LoRA sit upstream of all primitives. After Phase 1, every primitive is frozen, giving a parameter-level answer to the *where* question: skill s_i's atomic update remains (B_i, A_i).

Composite training introduces a rank-10 residual $R_i = D_i C_i$ for every skill, attached to the action expert's attention and feed-forward projections alongside P_i. The residual scaling factor is one; C_i is randomly initialized and D_i is initialized to zero. Thus, adding the residual initially preserves each branch's output under the same routing weights. Training subsequently adapts R_i while preserving P_i.

Composite Skill Labels. Composite demonstrations provide task-level instructions but no per-chunk atomic-skill labels, so PRISM trains the router with labels from a labeling oracle. Formally, the oracle $\mathcal{A}$ maps a demonstration $(s_1, a_1), \ldots, (s_n, a_n)$ to a skill-index sequence $\{z_t\}_{t=1}^{n}$, with $z_t \in \{1, \ldots, N\}$, consistent with the atomic skill taxonomy. These labels serve as cross-entropy targets for router training.

We derive frame-level skill labels from task identity and gripper events in demonstration actions. Task identity specifies the ordered skill pair, while gripper opening or closing events anchor the transition between skill phases. Task-dependent temporal offsets include approach and post-release motions in the corresponding phases. These labels provide skill-level supervision for the router; the complete labeling rules are given in Appendix D.

Prefix-Conditioned Routing and Skill Composition. The router decides, at each chunk boundary, which primitive–residual branches the action expert activates. Its input is a mask-aware mean-pool of the embedding-level prefix tokens:

$$e_t = [\mathrm{SigLIP}(I_t);\, \mathrm{Embed}_{\mathrm{Gemma}}(\ell)], \tag{3}$$

where I_t denotes the camera observations, SigLIP supplies the visual tokens, and $\mathrm{Embed}_{\mathrm{Gemma}}$ is the instruction-token embedding lookup before the language transformer. With input/padding mask $m_{t,s}$, the router applies LayerNorm and a two-layer MLP followed by softmax:

$$h_t = \frac{\sum_s m_{t,s} e_{t,s}}{\sum_s m_{t,s}}, \qquad \alpha(t) = \mathrm{softmax}(\mathrm{MLP}_\theta(\mathrm{LN}(h_t)))\,. \tag{4}$$

The router is initialized to uniform weights and recomputes the mixture for every action chunk.

For each adapted attention projection, the router combines the primitive and residual contributions as

$$y = W_0 x + \sum_{i=1}^{N} \alpha_i(t)\big(P_i x + R_i x\big). \tag{5}$$

For feed-forward layers, each skill branch evaluates its complete FFN, including its nonlinearities, before output mixing:

$$y_{\mathrm{FFN}} = \sum_{i=1}^{N} \alpha_i(t) F_i(x; W_0, P_i, R_i). \tag{6}$$

Here F_i contains the shared backbone weights and the skill's primitive and residual adapters. Mixing branch outputs preserves each branch's computation and avoids the cross-skill products introduced by blending LoRA factors. The same chunk-level weights govern skill composition throughout the action expert.

Two-Stage Optimization and Execution. Phase 1 learns the primitives from atomic demonstrations using task-specified one-hot routing and a flow-matching action objective [26]. Alongside the primitives, we train the visual encoder, shared language LoRA, action input/output projections, and time MLP; the language and action-expert backbone weights remain frozen. Phase 2 freezes the primitives and trains the residual adapters and router together with these shared adaptation modules on composite demonstrations and atomic replay. The objective is

$$\mathcal{L}_{\mathrm{Phase\,2}} = \mathcal{L}_{\mathrm{FM}} + 0.05\,\mathcal{L}_{\mathrm{route}}, \tag{7}$$

where $\mathcal{L}_{\mathrm{FM}}$ is the squared error between predicted and target flow velocities, and $\mathcal{L}_{\mathrm{route}}$ is the routing cross-entropy. We mask routing cross-entropy near labeled transitions while retaining the action objective throughout, allowing the policy to learn soft skill mixtures across phase boundaries. Training uses the router's soft weights for action generation, so the action objective and skill supervision jointly optimize routing. Dataset composition and the training schedule are given in Appendix A.

At execution time, composite tasks use the learned router to update skill weights for each action chunk. Atomic tasks use task-specified one-hot routing to execute the corresponding primitive–residual branch.

5 Experiments

We evaluate PRISM in simulation (Sections 5.1–5.4) and on a real robot (Section 5.5). The simulation experiments are designed to answer four questions: (i) does PRISM's primitive-bank-plus-router architecture outperform a standard $\pi_{0.5}$-LoRA fine-tune baseline on long-horizon composite tasks? (ii) does PRISM preserve atomic-skill performance through composite training relative to the same baseline and to a sample-level continual-learning variant (replay)? (iii) which design choices in PRISM (labeling oracle, router conditioning, output composition) actually matter? (iv) do these gains transfer to a physical robot?

5.1 Simulation Setup

Benchmark and skill setup. The RoboCasa365 [34] atomic dataset contains 3,016 human demonstrations across 28 tasks: Pick & Place (18 tasks, 1,939 demonstrations), Turn Lever (3 tasks, 322

Table 1: **Atomic and composite task success rate on RoboCasa365.** Atomic SR is the macro-average over 28 atomic tasks (18 PP + 3 Lever + 7 Close); Composite SR is the macro-average over 4 composite tasks. "Atomic-only" rows are stage-1 checkpoints; "atomic + comp." rows are the full two-stage fine-tune. PRISM bank only is the Phase-1 checkpoint without a fitted router. Δ columns report SR difference (pp) against the $\pi_{0.5}$-LoRA two-stage fine-tune (†).

Method	Atomic SR	Δ Atomic	Composite SR	Δ Composite
$\pi_{0.5}$ (zero-shot)	0.00	-17.86	0.00	-17.50
$\pi_{0.5}$-LoRA, atomic-only	48.04	$+30.18$	0.00	-17.50
$\pi_{0.5}$-LoRA, atomic + comp. †	17.86	0.00	17.50	0.00
$\pi_{0.5}$-LoRA + replay	49.82	$+31.96$	10.25	-7.25
PRISM, bank only (Phase 1)	50.71	$+32.85$	0.00	-17.50
PRISM (ours)	**56.43**	$+38.57$	**20.00**	$+2.50$

demonstrations), and Close (7 tasks, 755 demonstrations). These families define the three primitive slots. The composite split contains 415 demonstrations of LoadDishwasher, SnackSorting, ThawInSink, and WashLettuce. LoadDishwasher and SnackSorting transition from Pick & Place to Close; ThawInSink transitions from Pick & Place to Turn Lever; WashLettuce transitions from Turn Lever to Pick & Place in the demonstrations. All four tasks require cross-skill composition. Phase 2 combines these demonstrations with 20 replay episodes per atomic task (560 episodes), yielding 975 episodes and 454,758 frames. Appendix C.1 lists the complete task suite and data composition.

Baselines. We compare policies built on $\pi_{0.5}$: zero-shot inference, LoRA trained on atomic tasks, LoRA trained sequentially on atomic and composite tasks, and the replay-augmented two-stage LoRA baseline. The PRISM Phase 1 checkpoint isolates atomic primitive learning. Full PRISM adds skill-specific residuals and a per-chunk router during composite training. Table 1 compares atomic retention and composite success across these configurations.

5.2 Metrics and Protocol

We report atomic SR as the macro-average over 28 atomic tasks, with 20 trials per task, and composite SR as the macro-average over four composite tasks, with 100 trials per task. PRISM uses task-specified one-hot routing for atomic evaluation and learned per-chunk routing for composite evaluation. The full evaluation protocol is provided in Appendix C.

5.3 Main Results: Atomic and Composite Task Success

Higher atomic and composite success. Table 1 shows that PRISM achieves $\mathbf{56.43}\%$ atomic success and $\mathbf{20.00}\%$ composite success, exceeding the standard $\pi_{0.5}$-LoRA two-stage fine-tune (17.86% and 17.50%) by $\mathbf{38.57}$ and $\mathbf{2.50}$ percentage points. Relative to LoRA with replay (49.82% and 10.25%), PRISM improves atomic and composite success by 6.61 and 9.75 points, respectively, nearly doubling composite success over the replay baseline.

Atomic retention through composite training. Standard two-stage LoRA fine-tuning reduces atomic success from 48.04% to 17.86%, a 30.18-point drop. PRISM instead raises atomic success from 50.71% in Phase 1 to 56.43% after composite training, a 5.72-point gain, while keeping the primitive parameters fixed.

Composite adaptation of the primitive bank. PRISM's composite success rises from 0.00% in Phase 1 to 20.00% after Phase 2. Residuals, routing, and shared adaptation modules jointly turn the atomic foundation into a policy that executes the four composite tasks; the ablations isolate the contribution of the routing design.

5.4 Ablations

We ablate three core PRISM design choices in Table 2: (a) which oracle provides per-chunk skill labels for router training, (b) what input the router conditions on, and (c) how the router output is composed with the primitive bank. Each panel reports composite macro-average SR.

Table 2: **Three ablations on PRISM design choices, reported as composite macro-average SR.** (a) The composite skill labeling oracle of Section 4. (b) The router conditioning input (Section 4). (c) The router output and primitive composition mechanism.

(a) Labeling oracle

Oracle	Comp. Avg
Qwen2.5-VL	11.25
MLP (s_t, a_t)	13.75
Kinematic (ours)	**20.00**

(b) Router input

Input	Comp. Avg
Vision only	10.25
State only	7.50
VLM prefix (ours)	**20.00**

(c) Router output & composition

Composition	Comp. Avg
Hard one-hot	10.50
Soft, param-space	15.50
Soft, action-space	13.50
Soft, activation (ours)	**20.00**

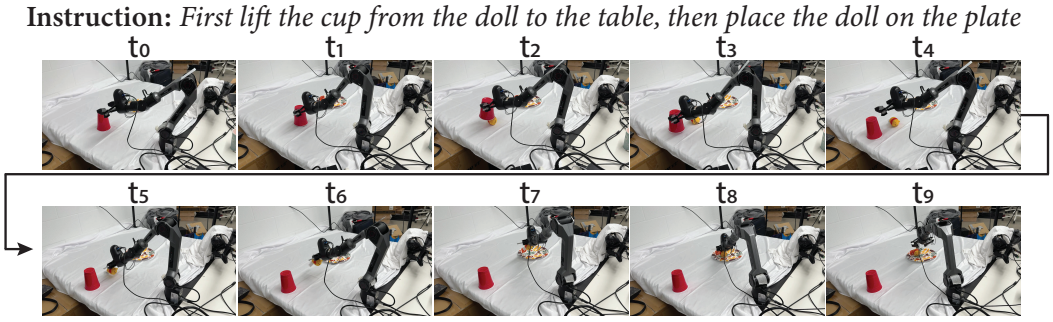


Figure 5: **Real-robot composite rollout under PRISM.** A successful inference on the Cup → Table → Doll → Plate composite task: the robot first lifts the cup off the doll (t_0–t_4), then picks up the doll and places it on the plate (t_5–t_9). The router supplies a per-chunk mixture $\alpha(t)$ over the bank that switches from the Cup → Table primitive to the Doll → Plate primitive mid-trajectory.

Labeling oracle (panel a). The oracle $\mathcal{A}$ is a swappable interface (§4); we compare our default kinematic rule against a Qwen2.5-VL [1] oracle on video frames and a learned MLP on (s_t, a_t). The kinematic rule outperforms both (**20.00** vs MLP 13.75 vs VLM 11.25): the VLM's transition placement varies under prompt and viewpoint changes, while the deterministic rule anchors skill transitions to task identity and gripper events.

Router conditioning (panel b). A chunk-level routing decision needs both task identity and progress within the task. Pooled vision features carry progress but no instruction identity (10.25); proprioceptive state is too low-dimensional to track phase transitions (7.50); only the VLM prefix h_t fuses both signals into a single contextual feature (**20.00**).

Router output and composition (panel c). *Hard one-hot* (10.50) imposes discrete branch selection across the cross-skill transitions.

Soft + parameter-space (15.50) mixes LoRA factors before applying the update. For a set of factors (B_i, A_i), the schematic product $(\sum_i \alpha_i B_i)(\sum_j \alpha_j A_j)$ introduces cross-skill terms $\alpha_i \alpha_j B_i A_j$ for $i \neq j$.

Soft + action-space (13.50) executes N policy forward passes and averages the resulting action chunks at the output.

Soft + activation-space (PRISM, **20.00**) blends primitive–residual contributions in attention projections and complete skill-specific FFN outputs, avoiding cross-skill LoRA-factor products within one action-expert forward pass.

5.5 Real-Robot Experiments

We evaluate PRISM on an AgileX PIPER 6-DoF arm with two atomic skills (Cup → Table, Doll → Plate) and one composite chaining them; gripper events in demonstration actions provide skill-phase supervision (Section 4). We collect 20 teleoperated demonstrations per task and roll out 20 trials per task with matched initial configurations; full hardware, task, and protocol details are in Appendix C. Figure 5 shows a successful PRISM rollout on the composite task.

Table 3: **Real-robot success rate on AgileX PIPER.** Two atomic skills (Cup → Table, Doll → Plate) and a Composite chaining them. Δ columns report absolute SR difference (pp) against the $\pi_{0.5}$-LoRA two-stage fine-tune (†). SR↑, Δ ↑.

	Cup→Table		Doll→Plate		Composite	
Method	SR	Δ	SR	Δ	SR	Δ
$\pi_{0.5}$ (zero-shot)	0.0	−25.0	0.0	−35.0	0.0	−45.0
$\pi_{0.5}$-LoRA, atomic-only	55.0	+30.0	75.0	+40.0	0.0	−45.0
$\pi_{0.5}$-LoRA, atomic + comp. †	25.0	0	35.0	0	45.0	0
$\pi_{0.5}$-LoRA + replay	50.0	+25.0	65.0	+30.0	50.0	+5.0
PRISM, bank only (Phase 1)	65.0	+40.0	80.0	+45.0	0.0	−45.0
PRISM (ours)	**65.0**	**+40.0**	**85.0**	**+50.0**	**55.0**	**+10.0**

Results. Table 3 mirrors the simulation pattern. The $\pi_{0.5}$-LoRA two-stage fine-tune loses 30–40 atomic points through composite training; replay partially recovers atomic but still trails on composite. PRISM preserves atomic SR through Phase 2 (Cup → Table $65\%\rightarrow65\%$, Doll → Plate $80\%\rightarrow85\%$) and reaches $\mathbf{55.0}\%$ composite—highest in every column.

6 Conclusion

We presented PRISM, a VLA adaptation framework that represents reusable skills as parameter-level primitives. PRISM exploits an empirical structural finding—per-skill LoRA updates span a near-N-dimensional, semantically clustered subspace that captures $\sim56\%$ of a composite-task LoRA's energy—and operationalizes it through a primitive bank with skill-specific residual adaptation, a per-chunk VLM-conditioned router, and a deterministic action-derived labeler. On RoboCasa365, PRISM achieves $\mathbf{56.43}\%$ atomic success after composite training and reaches $\mathbf{20}\%$ on long-horizon composite tasks, outperforming both two-stage fine-tuning and replay. Sequential addition of new skill families and scaling beyond three primitives are natural next steps for the bank-plus-router design.

Acknowledgments and Disclosure of Funding

This research was partially supported by NSF grants IIS-2525840, CNS-2432534, and ECCS-2514574. Bing Liu's work was supported in part by NSF grants IIS-2229876 and CNS-2225427. The views and conclusions expressed in this article are solely those of the authors and do not necessarily reflect those of the funding agencies or supporting organizations.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, 2025. URL `https://arxiv.org/abs/2502.13923`.

[2] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. PaliGemma: A versatile 3B VLM for transfer, 2024. URL `https://arxiv.org/abs/2407.07726`.

[3] 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 *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

[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*, 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Conference on Robot Learning (CoRL)*, 2023. URL https://arxiv.org/abs/2307.15818.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex 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 Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong 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 *Robotics: Science and Systems (RSS)*, 2023. URL https://arxiv.org/abs/2212.06817.

[7] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations (ICLR)*, 2019.

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems (RSS)*, 2023. URL https://arxiv.org/abs/2303.04137.

[10] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2401.06066.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.14314.

[12] 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 *International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2303.03378.

[13] William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research (JMLR)*, 23(120):1–39, 2022. URL https://arxiv.org/abs/2101.03961.

[14] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, et al. Gemma: Open Models Based on Gemini Research and Technology, 2024. URL https://arxiv.org/abs/2403.08295.

[15] Minghao Guo, Meng Cao, Jiachen Tao, Rongtao Xu, Yan Yan, Xiaodan Liang, Ivan Laptev, and Xiaojun Chang. GLaD: Geometric Latent Distillation for Vision-Language-Action Models, 2025. URL https://arxiv.org/abs/2512.09619.

[16] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP. In *International Conference on Machine Learning (ICML)*, 2019.

[17] 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 (ICLR)*, 2022.

[18] Suning Huang, Zheyu Aqa Zhang, Tianhai Liang, Yihan Xu, Zhehao Kou, Chenhao Lu, Guowei Xu, Zhengrong Xue, and Huazhe Xu. MENTOR: Mixture-of-Experts Network with Task-Oriented Perturbation for Visual Reinforcement Learning. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 26143–26161. PMLR, 2025. URL https://proceedings.mlr.press/v267/huang25av.html.

[19] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of Experts, 2024. URL https://arxiv.org/abs/2401.04088.

[20] 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 *Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

[21] 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 (PNAS)*, 114 (13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

[22] Yuheng Lei, Sitong Mao, Shunbo Zhou, Hongyuan Zhang, Xuelong Li, and Ping Luo. Dynamic Mixture of Progressive Parameter-Efficient Expert Library for Lifelong Robot Learning. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=MHVBrjS8cG.

[23] 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, 2024. URL https://arxiv.org/abs/2411.19650.

[24] Zhizhong Li and Derek Hoiem. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(12):2935–2947, 2018. URL https://arxiv.org/abs/1606.09282.

[25] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. OneTwoVLA: A Unified Vision-Language-Action Model with Adaptive Reasoning, 2025. URL https://arxiv.org/abs/2505.11917.

[26] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.02747.

[27] 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 (NeurIPS)*, 2023.

[28] Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, Chengkai Hou, Mengdi Zhao, KC alex Zhou, Pheng-Ann Heng, and Shanghang Zhang. HybridVLA: Collaborative Diffusion and Autoregression in a Unified Vision-Language-Action Model, 2025. URL https://arxiv.org/abs/2503.10631.

[29] 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 (ICLR)*, 2025. URL https://arxiv.org/abs/2410.07864.

[30] Tenglong Liu, Jianxiong Li, Yinan Zheng, Haoyi Niu, Yixing Lan, Xin Xu, and Xianyuan Zhan. Skill Expansion and Composition in Parameter Space. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2502.05932.

[31] Arun Mallya and Svetlana Lazebnik. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[32] Xiaofeng Mao, Gabriele Giudici, Claudio Coppola, Kaspar Althoefer, Ildar Farkhatdinov, Zhibin Li, and Lorenzo Jamone. DexSkills: Skill Segmentation Using Haptic Data for Learning Autonomous Long-Horizon Robotic Manipulation Tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. URL https://arxiv.org/abs/2405.03476.

[33] Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. QueST: Self-Supervised Skill Abstractions for Learning Continuous Control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2407.15840.

[34] Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2603.04356.

[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, Loic 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, 2025. URL https://arxiv.org/abs/2503.14734.

[36] 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. In *Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2405.12213.

[37] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. URL https://arxiv.org/abs/2310.08864.

[38] 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. In *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2501.09747.

[39] 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 *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2501.15830.

[40] Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient Diffusion Transformer Policies with Mixture of Expert Denoisers for Multitask Learning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2412.12953.

[41] Ralf Römer, Yi Zhang, Yuming Li, and Angela P. Schoellig. CLARE: Continual Learning for Vision-Language-Action Models via Autonomous Adapter Routing and Expansion. *IEEE Robotics and Automation Letters*, 11(7):8331–8338, 2026. doi: 10.1109/lra.2026.3693992. URL https://doi.org/10.1109/lra.2026.3693992.

[42] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks, 2016. URL https://arxiv.org/abs/1606.04671.

[43] Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-Ended Instruction Following with Hierarchical Vision-Language-Action Models. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 54919–54933. PMLR, 2025. URL https://proceedings.mlr.press/v267/shi25d.html.

[44] Robin Strudel, Alexander Pashevich, Igor Kalevatykh, Ivan Laptev, Josef Sivic, and Cordelia Schmid. Learning to combine primitive skills: A step towards versatile robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. URL https://arxiv.org/abs/1908.00722.

[45] Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. LOTUS: Continual Imitation Learning for Robot Manipulation Through Unsupervised Skill Discovery. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. URL https://arxiv.org/abs/2311.02058.

[46] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. DexVLA: Vision-Language Model with Plug-In Diffusion Expert for General Robot Control. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 3094–3114. PMLR, 2025. URL https://proceedings.mlr.press/v305/wen25b.html.

[47] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. TinyVLA: Toward Fast, Data-Efficient Vision-Language-Action Models for Robotic Manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025. doi: 10.1109/lra.2025.3544909. URL https://doi.org/10.1109/lra.2025.3544909.

[48] Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. DiffusionVLA: Scaling Robot Foundation Models via Unified Diffusion and Autoregression. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 66558–66574. PMLR, 2025. URL https://proceedings.mlr.press/v267/wen25g.html.

[49] Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, Cewu Lu, and Wenqiang Zhang. ForceVLA: Enhancing VLA Models with a Force-aware MoE for Contact-rich Manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.22159.

[50] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning (ICML)*, 2017.

[51] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid Loss for Language Image Pre-Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://arxiv.org/abs/2303.15343.

[52] Xuanran Zhai, Zekai Huang, Longyan Wu, Qianyou Zhao, Qiaojun Yu, Jieji Ren, Ce Hao, and Harold Soh. SkillVLA: Tackling Combinatorial Diversity in Dual-Arm Manipulation via Skill Reuse. In *Proceedings of Robotics: Science and Systems*, 2026. doi: 10.15607/RSS.2026.XXII.082. URL https://www.roboticsproceedings.org/rss22/p082.html.

[53] Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, et al. AtomicVLA: Unlocking the Potential of Atomic Skill Learning in Robots. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2603.07648.

[54] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Robotics: Science and Systems (RSS)*, 2023. URL https://arxiv.org/abs/2304.13705.

[55] Tony Z. Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Seyed Kamyar Seyed Ghasemipour, Chelsea Finn, and Ayzaan Wahid. ALOHA Unleashed: A Simple Recipe for Robot Dexterity. In *Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 1910–1924. PMLR, 2025. URL https://proceedings.mlr.press/v270/zhao25b.html.

[56] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D Vision-Language-Action Generative World Model. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2403.09631.

A Training Pipeline

Backbone configuration. PRISM is built on $\pi_{0.5}$ [3]: vision encoder ϕ_{V} is the SigLIP encoder [51] from PaliGemma [2]; language model ϕ_{L} is Gemma [14] with a shared rank-16 LoRA L on attention and feed-forward sublayers; action expert π_{A} is Gemma, into which three per-skill primitive LoRAs (rank-32) are injected over the action-expert weight $W_{A,0}$. In Phase 2, each primitive is augmented with a trainable rank-10 residual.

PRISM is trained by a two-phase pipeline.

Phase 1 (atomic primitive learning, $T_1 = 30$K steps). Starting from pretrained $\pi_{0.5}$ weights, each atomic demonstration is hard-assigned to its skill slot i (one-hot α), routing the whole episode through (B_i, A_i) alone. We train the bank $\{(B_i, A_i)\}_{i=1}^{3}$ through the selected primitive, full-fine-tune ϕ_{V}, and adapt the shared rank-16 language LoRA L, action input/output projections, and time MLP. Both Gemma backbones, language token embeddings, and language normalization layers are frozen. The atomic dataset is described in Appendix C.1. Training uses batch size 32, 1,000 warmup steps, and a cosine learning-rate schedule from 2.5×10^{-5} to 2.5×10^{-6}. Phase 2 starts from the final Phase-1 checkpoint.

Phase 2 (joint composite training, $T_2 = 20$K steps). We warm-start from Phase 1 and train on D_{replay}, which combines 415 composite demonstrations with $K = 20$ replay episodes per atomic task (560 episodes across 28 tasks; Appendix C.1). Frames are sampled uniformly from the merged dataset, without class balancing. Trainable parameters are ϕ_{V}, L, r_θ, the three rank-10 skill residuals, action input/output projections, and time MLP; the primitive bank and both Gemma backbones remain frozen.

With z_t from the labeling oracle (Section 4), the loss is $\mathcal{L}_{\text{FM}} + 0.05\mathcal{L}_{\text{route}}$ (Equation 7). Routing cross-entropy excludes frames within 100 frames on either side of a labeled transition, while the flow-matching objective uses all sampled frames. Action prediction uses soft router weights, so the action objective also updates the router. The routing loss propagates into ϕ_{V} through the embedding-level prefix; L receives gradients through the action objective.

Training uses batch size 32, 500 warmup steps, and a cosine learning-rate schedule from 2.5×10^{-5} to 2.5×10^{-6}. Evaluation uses the final Phase-2 checkpoint.

Why freeze the bank in Phase 2. Jointly updating the primitive bank allows composite training to modify the atomic updates, breaking the parameter-isolation contract. Freezing the bank preserves the atomic updates while skill-specific residuals and shared modules adapt to composite behavior. The router combines the primitive–residual branches, while $\mathcal{L}_{\text{CE}}$ against z_t anchors r_θ to the three skill families. Replay atomic frames provide single-skill examples that anchor the router on the atomic distribution, while composite frames train the residuals, router, and shared modules on phase-switching trajectories.

Lifelong extension. The bank supports adding a new skill through an additional primitive–residual branch and a corresponding router output, while preserving the existing primitive parameters.

Existing primitives are never updated after their own Phase-1 training, so the bank's frozen-after-Phase-1 property is preserved across additions.

B Limitations

Skill taxonomy and supervision. PRISM uses predefined skill families, and its action-derived labeling rules use task identity, gripper events, and temporal offsets to assign skill phases. These rules assume that the relevant transitions can be localized from demonstration actions; tasks without such event cues require a different labeling mechanism. The router-supervision interface accommodates alternative labelers, including the learned classifier and VLM evaluated in Section 5.4.

Evaluation scope and retention. The policy evaluation studies atomic-skill retention through composite adaptation with a fixed bank: three primitives and four composite tasks in simulation, and two primitives and one composite task on the real robot. The real-robot evaluation uses closely matched initial configurations. The independent $N = 6$ and $N = 11$ banks in Section 3 establish the measured primitive geometry; sequential skill acquisition and policy evaluation with larger banks remain extensions of the present setting. Freezing the primitive bank preserves its parameters, while skill residuals and shared modules continue to adapt. Behavioral retention is therefore measured by atomic-task success after composite training, using task-specified routing; composite evaluation uses the learned router.

C Evaluation Protocol

This appendix documents the evaluation protocol in detail, complementing the headline metrics described in Section 5.2.

Per-task rollouts. The RoboCasa365 atomic suite uses 20 trials per task and the composite suite uses 100 trials per task. PRISM uses seed base 17 for atomic evaluation and 31 for composite evaluation. The physical-robot study uses 20 trials per task.

Success criterion. For RoboCasa365 atomic and composite tasks, success is determined by the simulator's built-in task-completion checker. For the real-robot setup, success is judged by a human

evaluator (Cup $\rightarrow$ Table: cup placed on the table surface without dropping; Doll $\rightarrow$ Plate: doll placed on the plate without dropping; Composite: both succeed in sequence).

Aggregation. Atomic SR averages success equally over 28 tasks, and composite SR averages success equally over four tasks. Equal trial counts within each suite make these macro-averages equal to the pooled success fractions.

Execution modes. Atomic evaluation uses task-specified routing, activating the corresponding primitive for Phase 1 and the primitive–residual branch for the complete policy. Composite evaluation of the complete policy recomputes the learned router weights at each replanning step.

C.1 RoboCasa365 Dataset Details

Dataset summary. All training demonstrations are human demonstrations from RoboCasa365, recorded at 20 Hz. The atomic split contains 3,016 demonstrations and 687,997 frames across 28 tasks in three skill families: Pick & Place, Turn Lever, and Close. Table 4 gives the complete inventory. LoadDishwasher, SnackSorting, ThawInSink, and WashLettuce supply 415 composite demonstrations and 327,608 frames.

Table 4: **RoboCasa365 atomic dataset.** Per-task episode and frame counts for the 28 atomic tasks used in PRISM's training and evaluation.

Task	Episodes	Frames
Pick & Place (18 tasks)		
PickPlaceCabinetToCounter	106	20,201
PickPlaceCounterToBlender	106	38,892
PickPlaceCounterToCabinet	108	24,225
PickPlaceCounterToDrawer	107	28,225
PickPlaceCounterToMicrowave	110	42,012
PickPlaceCounterToOven	111	32,014
PickPlaceCounterToSink	108	22,410
PickPlaceCounterToStandMixer	107	25,467
PickPlaceCounterToStove	108	24,039
PickPlaceCounterToToasterOven	108	24,313
PickPlaceDrawerToCounter	103	31,819
PickPlaceFridgeDrawerToShelf	106	26,396
PickPlaceFridgeShelfToDrawer	111	27,047
PickPlaceMicrowaveToCounter	112	38,729
PickPlaceSinkToCounter	108	26,397
PickPlaceStoveToCounter	109	23,003
PickPlaceToasterOvenToCounter	106	19,323
PickPlaceToasterToCounter	105	26,907
subtotal	1,939	501,419
Turn Lever (3 tasks)		
TurnOnSinkFaucet	107	23,795
TurnOffSinkFaucet	106	12,309
TurnSinkSpout	109	11,130
subtotal	322	47,234
Close (7 tasks)		
CloseCabinet	105	27,754
CloseDishwasher	111	15,781
CloseDrawer	110	15,670
CloseFridge	106	26,888
CloseFridgeDrawer	111	14,946
CloseMicrowave	105	20,075
CloseOven	107	18,230
subtotal	755	139,344
Total	**3,016**	**687,997**

Composite demonstrations. LoadDishwasher requires placing a cup and bowl in the dishwasher and closing its door; both placement interactions belong to the Pick & Place phase. SnackSorting requires placing a bar in a bowl and closing a drawer. ThawInSink requires placing the instructed object in the sink and then turning on the faucet. WashLettuce asks the robot to wash lettuce with running water; its demonstration-labeling rule assigns Turn Lever followed by Pick & Place. Task identity specifies the skill order, and gripper events locate the transition.

Composite task	Episodes	Frames	Mean frames / episode
LoadDishwasher	100	84,383	843.83
SnackSorting	100	82,518	825.18
ThawInSink	111	81,459	733.86
WashLettuce	104	79,248	762.00
Total	415	327,608	789.42

Phase 2 mixture. The 415 composite demonstrations are combined with 560 atomic replay demonstrations (20 for each task), totaling 975 episodes. Replay contributes $127{,}150$ frames; the merged dataset contains $454{,}758$ frames. Training samples naturally by frame without class balancing, so composite and atomic replay account for 72.04% and 27.96% of the frame population.

C.2 Real-Robot Setup

Hardware. The platform is an AgileX PIPER 6-DoF arm with a parallel-jaw gripper. Observations are RGB images from a single wrist-mounted fisheye camera, together with the arm's joint state and gripper aperture. The action expert produces chunks of horizon $H = 10$ at 30 Hz, giving each chunk a lookahead of $H/f \approx 0.33$ s.

Tasks. We define two atomic skill families and one composite task.

- **Cup → Table.** A cup is initially placed on top of a doll; the robot lifts the cup off the doll and places it on the table surface.
- **Doll → Plate.** A doll is initially on the table; the robot picks up the doll and places it on a target plate.
- **Composite (Cup → Table → Doll → Plate).** The composite chains the two atomic skills sequentially.

The composite task requires a primitive switch mid-trajectory once the cup has been placed and the robot turns to approach the doll. Each pick event corresponds to an open-to-close gripper edge in the action stream, providing an event anchor for the action-derived labeling oracle introduced in Section 4.

Demonstrations and evaluation protocol. We collect 20 teleoperated demonstrations per task (60 episodes total: 40 atomic and 20 composite). Each method is evaluated for 20 trials per task using closely matched initial configurations. We report mean success rate over the 20 trials; success is judged by a human evaluator under the criteria stated above (Cup → Table / Doll → Plate / Composite). We compare PRISM against the same four $\pi_{0.5}$ baselines as in simulation (zero-shot; LoRA atomic-only; LoRA two-stage atomic + composite; LoRA + replay). PRISM uses $N = 2$ primitive adapters and the kinematic labeling oracle, with atomic training followed by composite training.

D Kinematic Labeling Oracle: Full Rule

This appendix specifies the action-derived rule used to instantiate the labeling oracle $\mathcal{A}$ in Section 4.

Composite skill labeling rules. The labeler combines task identity with the gripper-command channel $g_t = a_t[11]$, treating $g_t > 0.5$ as closed. Each composite demonstration is represented by two ordered skill phases separated by one transition. Multiple grasp-and-release interactions within a phase share the same skill label. Atomic demonstrations receive their task's skill label throughout.

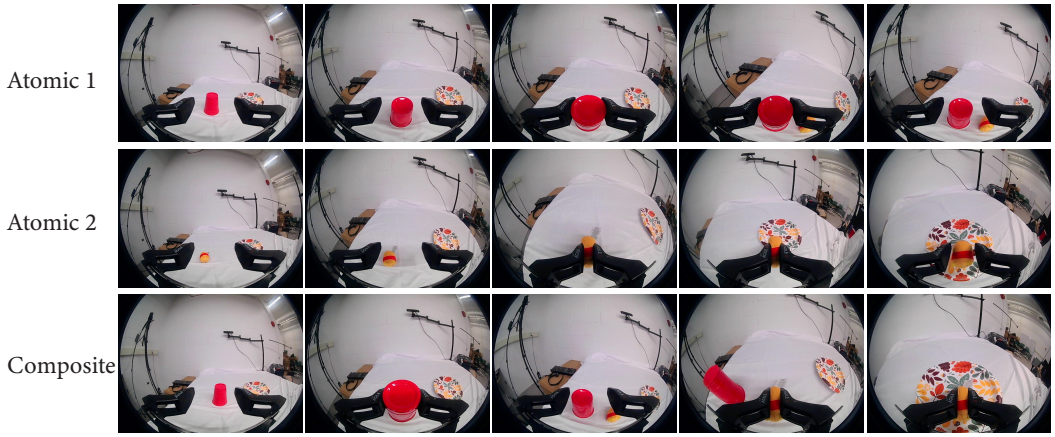


Figure 6: **Real-robot training data: fisheye observations.** Representative frames from the wrist-mounted fisheye RGB stream used for teleoperated training, sampled left-to-right along each trajectory. **Top:** *Cup* → *Table* (Atomic 1) – the red cup is lifted off the doll and placed on the table. **Middle:** *Doll* → *Plate* (Atomic 2) – the doll is picked up and placed on the patterned plate. **Bottom:** the *Composite* chains the two atomic skills in sequence.

Post-release transitions. LoadDishwasher and SnackSorting follow Pick & Place → Close, while ThawInSink follows Pick & Place → Turn Lever. Let R be the last closed-to-open gripper event. The transition is placed at $T = R + 30$, adding 1.5 seconds at the 20 Hz demonstration rate before switching to the second skill.

Pre-grasp transitions. WashLettuce follows Turn Lever → Pick & Place. Let G be the last open-to-closed gripper event. The transition is shifted earlier according to $T = G - \min(210, 0.35G)$, including the approach to the grasp in the Pick & Place phase. The maximum backshift is 210 frames (10.5 seconds), capped at 35% of the elapsed frames preceding the grasp event.

Label assignment and supervision. Frames before the transition receive the first skill label, and the subsequent phase receives the second. If the required gripper event is absent, the entire demonstration receives the first skill label. Routing cross-entropy excludes frames within 100 frames (5 seconds) of the transition, while all sampled frames contribute to flow matching. The labels supervise the router during training; composite-task inference uses its predicted soft weights.

E Vision-Language-Action Models

This appendix collects the broader VLA backbone literature. Vision-Language-Action (VLA) models combine pretrained vision-language representations with robot action prediction, enabling language-conditioned manipulation across diverse tasks [3–6, 12, 20, 35, 36]. Recent systems improve this substrate through larger and more diverse robot data [37]; alternative action representations including autoregressive tokens, diffusion policies, flow matching, and chunked action prediction [9, 28, 29, 38, 54, 55]; and stronger spatial, hierarchical, or reasoning-augmented control modules [15, 23, 25, 39, 43, 46–48, 56]. These works primarily improve the VLA backbone, action decoder, data scale, or reasoning interface. In contrast, PRISM studies how a pretrained VLA should be *adapted*: rather than absorbing each new behavior into one shared update, PRISM gives reusable skills parameter-level identity and learns to route among them during long-horizon execution.

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 empirical geometry analysis establishes composite–primitive alignment. PRISM implements reuse through a frozen primitive bank, trainable residual adapters, and prefix-conditioned routing. The simulation evaluation covers three primitives, 28 atomic tasks, and four composite tasks, with 56.43% atomic and 20.00% composite success (Table 1). The physical evaluation tests atomic retention and sequential composition on AgileX PIPER.

   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: Appendix B states the predefined skill taxonomy, the two-phase gripper-event labeling assumption, the atomic-to-composite training sequence, the distinct atomic and composite routing protocols, and the matched initial conditions of the physical evaluation.

   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 contain formal theorems. The participation ratio (Eq. 1) and projected energy ratio (Eq. 2) are standard definitions used as empirical measurements, not theoretical claims requiring proof.

   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: Appendix A specifies primitive, residual, and language LoRA ranks of 32, 10, and 16; batch size 32; 30K and 20K training steps; warmups of 1,000 and 500 steps; and the learning-rate schedule. Equation 7 gives the action and routing objectives. Appendix D specifies task-conditioned labels and the boundary mask. Appendix C.1 gives the 3,016 atomic demonstrations, the 415 composite plus 560 replay demonstrations, and evaluation protocols.

   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: [No]

   Justification: We are preparing the code and accompanying data-access and reproduction instructions for public release at `https://github.com/tlxhlll/PRISM-VLA`. The repository has been created; the implementation and reproduction materials are not yet available.

   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: The PRISM training configuration is given in Appendix A: 30K atomic and 20K composite updates, 20 replay episodes per atomic task, natural frame sampling, and routing loss weight 0.05. Appendix C specifies the datasets, evaluation checkpoints, trial counts, seeds, and execution modes.

   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: [No]

Justification: The reported PRISM simulation result uses one trained checkpoint, evaluated with 560 atomic trials and 400 composite trials. These trials measure rollout success for that checkpoint; they do not estimate variability across independent training runs. The random-rotation control reports mean and standard deviation over eight rotations per checkpoint. Standard deviation is a dispersion measure, not a confidence interval.

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: All training and evaluation in this paper was performed on $4\times$ NVIDIA RTX 6000 Ada Generation GPUs (48 GB VRAM each). A single complete PRISM run (Phase 1 30K AdamW steps + Phase 2 20K AdamW steps at peak LR 2.5×10^{-5}) takes approximately 12 hours wall-clock. The full experimental campaign in this paper, including the $\pi_{0.5}$-LoRA baselines, the experience-replay variant, the three ablation panels (oracle, router input, router composition), and the real-robot teleoperation/evaluation, used approximately $1{,}000$ GPU-hours in aggregate.

   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 work involves only simulated manipulation tasks and lab-scale teleoperated demonstrations collected by the authors; it does not involve human subjects, scraped or sensitive data, or deployments with privacy or fairness implications. We have reviewed the NeurIPS Code of Ethics and the research conforms in every respect.

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: Improved adaptation of VLA models to long-horizon manipulation has positive applications in assistive robotics, household automation, and industrial pick-and-place. Negative externalities are indirect: more capable manipulation policies could accelerate labor displacement in certain task domains, and any pretrained VLA backbone (including the $\pi_{0.5}$ family we build on) could in principle be repurposed beyond the tasks validated here. PRISM itself is a parameter-efficient adaptation framework and does not introduce new pretrained models or scraped datasets; broader concerns around large-scale VLA deployment are inherited from the upstream backbone rather than introduced by this work.

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 dual-use models, scraped datasets, or generative content. The released artifact is a training framework operating on top of the pretrained $\pi_{0.5}$ backbone (released by Physical Intelligence under their own license) and the RoboCasa365 simulator.

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: [No]

Justification: The original creators of the third-party models, code, and datasets are credited through citations. This paper does not explicitly enumerate the licenses and terms of use for all assets.

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: [No]

Justification: The release repository is `https://github.com/tlxhlll/PRISM-VLA`. We are preparing the PRISM implementation and documentation for training, evaluation, and dataset preparation; the release materials are not yet available.

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 research with human subjects. Real-robot teleoperation demonstrations were collected by the authors in a controlled lab setting.

    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: As stated in the previous item, no human-subjects research is performed in this paper, so IRB review does not apply.

    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: [Yes]

    Justification: PRISM uses the pretrained language model in $\pi_{0.5}$ for action conditioning and adapts its shared language LoRA during training (Section 3.1 and Appendix A). The router reads pooled visual and instruction embeddings before the language transformer (Section 4). Its default supervision uses deterministic, task-conditioned gripper-event rules rather than LLM-generated labels (Section 4). Qwen2.5-VL is used as an alternative labeling oracle in the Section 5.4 ablation.

    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.