# LEARNING LATENT ATOMIC MOVEMENTS FOR GENERALIZABLE AND ADAPTABLE ROBOT POLICIES

**Anonymous authors**
Paper under double-blind review

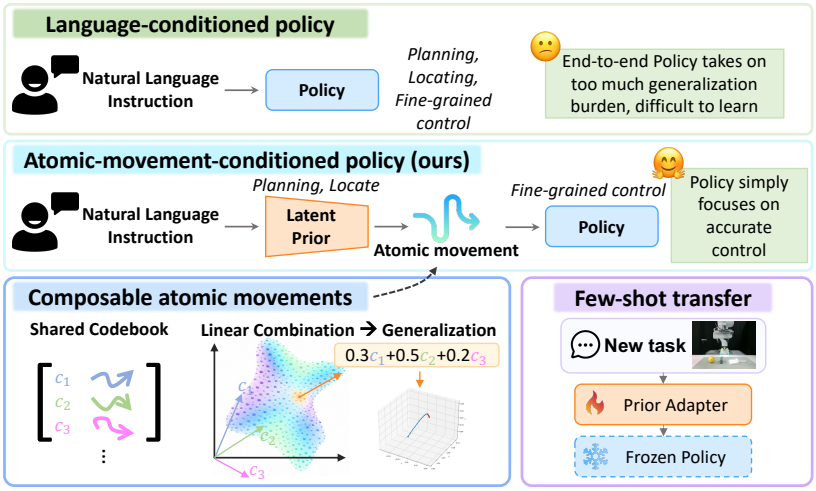



Figure 1: Current language-conditioned end-to-end robot policies must jointly perform planning, localization, and precise control, even with limited training data. We introduce composable atomic movements that capture coarse direction, replacing high-level language. This decomposition also enables modular transfer: adapting to a novel task requires tuning only a lightweight prior adapter.

## ABSTRACT

Adapting robot policies from limited demonstrations is challenging when task-dependent movement selection and precise control are learned jointly. We propose an *atomic-movement interface* that enables adaptation through movement selection while keeping the policy fixed. A latent action model learns compact movement representations as sparse combinations of shared codebook entries, supervised by action reconstruction. A latent prior predicts these representations from the instruction and observation history, and an observation-conditioned policy generates the corresponding actions. To encourage the policy to follow the movement condition, we introduce *Latent Override*, which pairs an observation with a compatible donor movement and its action target during training. After learning this interface, adaptation updates only the lightweight prior adapter while keeping the movement representation and policy fixed. Using Cosmos-Policy as the policy backbone, our method achieves 65.5% average success across ten real-world base tasks. Under few-shot adaption setting, it achieves 58.33% average success across six adaptation tasks, compared with 50.0% for Cosmos-Policy and 55.0% for $\pi_{0.5}$ under full-model fine-tuning, with lower measured adaptation time and GPU requirements. These results support reusing a learned movement interface and a fixed policy for few-shot robot policy adaptation.

## 1 INTRODUCTION

The goal of generalist robot learning is to build a single policy that can understand open-ended instructions and execute accurate physical actions. Recent vision–language–action (VLA) models



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

transfer semantic knowledge from large-scale vision–language pretraining to robot control (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024). World–action models (WAMs) take a complementary approach by adapting pretrained video models to predict robot actions and future observations (Kim et al., 2026; Ye et al., 2026; Yuan et al., 2026). Despite this progress, robot demonstrations are still much more expensive and difficult to collect than image and language data, even with large collaborative datasets (O'Neill et al., 2024). As a result, these models have strong visual and language priors but receive much less supervision for learning diverse robot movements.

This limited supervision must bridge a substantial gap between task instructions and precise robot trajectories. An instruction such as "stack the yellow cup into the gray cup" specifies a desired outcome, but leaves the short-term movements needed to achieve it implicit. An end-to-end policy must learn both which movement advances the task and how to execute it accurately in the current scene. Coupling these responsibilities can make adaptation costly, since learning a new task may require updating the full policy even when its low-level movements could be reused. This motivates separating task-dependent movement selection from observation-dependent execution.

An intermediate interface can make this separation explicit. Trajectory sketches provide spatial guidance (Gu et al., 2024), while visual traces supply motion context (Zheng et al., 2024). When the interface specifies a future path, the upstream model must predict its physical realization, which can be difficult to learn from a few demonstrations. Learned latent actions and skill representations offer a compact alternative and have already shown value for transfer (Ye et al., 2024; Chen et al., 2024; Mete et al., 2024; Chen et al., 2025). Building on this direction, we seek a shared movement representation that both conveys motion guidance and allows a separately trained policy to be reused across tasks. Its usefulness depends on two properties: it must encode motion information that can be reused across tasks, and the policy must respond to the movement it specifies.

As shown in Figure 1, we introduce an **atomic movement**: a compact latent representation of a short action chunk. A latent action model (LAM) encodes demonstrated video–action chunks into tokens formed from sparse combinations of shared codebook entries. At deployment, a latent prior predicts these tokens from the instruction and observation history, replacing the policy's language condition. To encourage the policy to follow the tokens despite correlated visual cues, we introduce **Latent Override**: during training, we retain an observation but substitute the latent and action target from a donor chunk with a similar initial configuration and a different movement. These examples encourage the policy to use the supplied movement to guide action generation.

Our contributions are threefold:

- We propose an atomic-movement interface that represents action chunks through linear combinations of shared codebook entries, separating task-dependent movement selection from observation-conditioned execution.
- We introduce Latent Override to train a policy to respond to the movement interface by pairing retained observations with compatible donor movements and action targets.
- We demonstrate few-shot adaptation by updating only the prior adapter while keeping the LAM, codebook, and policy fixed. Across six real-world tasks with 6–8 demonstrations per task, our method achieves 58.33% average success, compared with 50.0% for Cosmos-Policy and 55.0% for $\pi_{0.5}$ under full-model fine-tuning. Adaptation takes 97 minutes on one GPU, compared with 200 and 124 minutes on four GPUs for the respective baselines, retaining competitive task success with lower adaptation cost.

## 2 Related Works

### 2.1 Latent Actions in Games and Video World Models

Latent action learning recovers compact control variables from observed transitions, enabling learning from videos without action annotations. In games and video world modeling, discrete and continuous latent actions support interactive generation, action transfer, and adaptation across heterogeneous environments (Bruce et al., 2024; Gao et al., 2025). However, visual reconstruction alone does not ensure consistent action semantics across scenes. Olaf-World (Jiang et al., 2026) addresses this ambiguity through sequence-level control-effect alignment, anchoring latent actions to temporal feature differences from a frozen video encoder. SWIRL (Qiu et al., 2026) further explores learning

from state-only sequences by alternating updates to forward world models and inverse dynamics models. These approaches establish latent actions as useful controls for state prediction. Our work instead focuses on robot policy adaptation, grounding a chunk-level movement interface in demonstrated actions and training an observation-conditioned WAM to execute it.

### 2.2 Intermediate Interfaces for Robot Learning

Explicit visual and geometric interfaces connect high-level reasoning to physical control through marked affordances (Liu et al., 2024), trajectory sketches and 2D paths (Gu et al., 2024; Li et al., 2025), or 3D keypoint constraints and waypoints (Huang et al., 2024; Hwang et al., 2026). Historical visual traces provide complementary motion context (Zheng et al., 2024). These representations offer interpretable spatial guidance and reduce the policy's reasoning burden. However, their accuracy depends on the points, paths, or constraints supplied upstream. Lifting 2D paths into 3D can introduce depth ambiguity, motivating depth-aware trajectory guidance (Hwang et al., 2026). Even coarse sketches require the planner to specify a spatial realization of the intended movement, while geometric interfaces introduce dependencies on point localization or depth estimation. This motivates learning a compact interface that conveys movement information without requiring the upstream model to predict an explicit geometric plan.

Learned latent representations provide such an alternative. Video-derived action and motion tokens enable policy pretraining before adaptation to robot control (Ye et al., 2024; Chen et al., 2024), while quantized skill representations support few-shot transfer, including with a frozen decoder (Mete et al., 2024). More closely related, villa-X (Chen et al., 2025) grounds latent actions through visual and proprioceptive prediction and factorizes latent planning and robot action generation, with its action expert also receiving vision–language features. Other methods enrich policy representations through state-aware grounding and latent memory (Wang et al., 2025; Qu et al., 2026), or incorporate predictive dynamics through latent visual subgoals and shared transition–action representations (Chen et al., 2026; Lin et al., 2026). These developments demonstrate the value of latents for transfer, physical grounding, memory, and foresight. However, these roles do not by themselves establish an interface that reliably redirects a separately trained, observation-conditioned WAM. Our focus is therefore on both representing reusable movements and training the downstream policy to follow them.

A remaining challenge is that correlated visual context can let a policy bypass its latent condition, as also discussed in QueST's observation-conditioning ablation (Mete et al., 2024). Our method, on the other hand, learns atomic movements as sparse combinations of shared codebook entries from video–action chunks. Proposed Latent Override mechanism encourages the WAM to respond to the supplied movement condition. This combination supports adaptation by updating only the latent prior while keeping the LAM, codebook, and WAM fixed. It separates task-dependent movement selection from observation-dependent execution, allowing the pretrained policy to be reused without requiring an explicit waypoint plan.

## 3 Method

**Problem formulation.** Let $o_t$ denote the visual observation at time $t$, $\ell$ a language instruction, and $\mathbf{a}_t = (a_t, \dots, a_{t+H-1})$ an action chunk of horizon $H$. The policy predicts an action chunk using the instruction and available observations. During training, each demonstration also provides the corresponding observation chunk $\mathbf{o}_t^+ = (o_t, \dots, o_{t+H})$. We separate this prediction into movement selection and observation-conditioned execution, linked by an atomic-movement representation.

**Atomic movement: definition.** Let $\mathcal{C} = \{c_k\}_{k=1}^K$ be a shared codebook with entries $c_k \in \mathbb{R}^d$. We define an *atomic movement* as an ordered tuple of $M$ latent tokens representing one action chunk,

$$z_t = (z_{t,1}, \dots, z_{t,M}) \in \mathbb{R}^{M \times d}, \qquad z_{t,m} = \sum_{k=1}^{K} \alpha_{t,m,k} c_k,$$
$$\alpha_{t,m,k} \geq 0, \qquad \sum_{k=1}^{K} \alpha_{t,m,k} = 1, \qquad \|\alpha_{t,m}\|_0 \leq S. \tag{1}$$

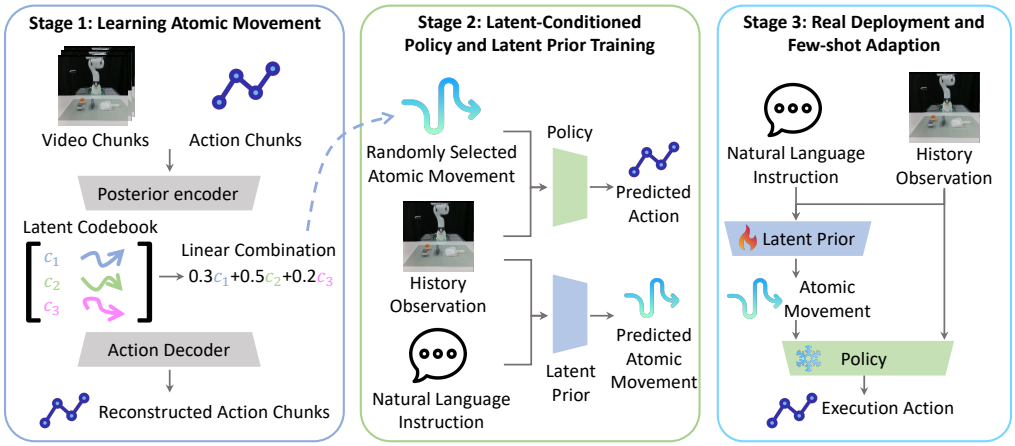



Figure 2: Overview of the atomic-movement framework. **Left**: the LAM learns a shared codebook through action reconstruction. **Middle**: the policy learns to execute latent-conditioned movements, and the prior learns to predict their representations from instructions and observation history. **Right**: the prior supplies the movement condition for policy execution; **few-shot adaptation updates only the prior adapter** while keeping the LAM, codebook, and policy fixed.

Here, $\alpha_{t,m}$ is the vector of codebook weights for token $m$, and $1 \le S \le K$ limits the number of nonzero weights. The full tuple is the unit selected by the prior and supplied to the policy at the action-chunk timescale. Its movement content is grounded by an action decoder trained to reconstruct the demonstrated chunk from $z_t$. The policy additionally uses the current observation to generate the trajectory for execution. Thus, the interface carries learned movement information while leaving action generation conditioned on the current scene.

**Learning and execution.** A training-time LAM posterior obtains the atomic movement from the complete demonstrated chunk and instruction:

$$z_t \sim q_\phi\big(z \mid \mathbf{o}_t^+, \mathbf{a}_t, \ell\big)\,. \tag{2}$$

At inference time, future observations and actions are unavailable. A latent prior predicts the atomic movement from a short observation history $\mathbf{h}_t$ ending at time $t$ and the instruction. The policy then predicts the action chunk from the current observation and this latent:

$$\hat{z}_t \sim p_\psi(z \mid \mathbf{h}_t, \ell)\,, \qquad \hat{\mathbf{a}}_t \sim \pi_\theta(\mathbf{a} \mid o_t, \hat{z}_t)\,. \tag{3}$$

Figure 2 summarizes representation learning, training of the policy and prior, and deployment with few-shot adaptation. We first learn and freeze the LAM and codebook (Section 3.1), then train the policy to follow this interface (Section 3.2) and the prior to predict it from available observations (Section 3.3). New tasks are learned by adapting only the prior adapter.

### 3.1 Learning Atomic Movements with a Latent Action Model

The LAM grounds the atomic-movement representation in trajectory demonstrations. It consists of a posterior encoder $q_\phi$, the shared codebook $\mathcal{C}$, and an action decoder $g_\omega$. The encoder takes the observation chunk, action chunk, and instruction as input. The observations provide context for the demonstrated motion, and the actions provide direct supervision for its representation. The encoder maps these inputs to the $M$ tokens defined in Eq. 1.

Sparse code selection lets action chunks share codebook entries while using different combinations of them. For token $m$, the encoder predicts scores $r_{t,m} \in \mathbb{R}^K$ and retains the $S$ highest-scoring entries. Their nonnegative weights are normalized over the selected set, with all other weights set to zero:

$$z_{t,m} = \sum_{k \in \mathrm{TopS}(r_{t,m})} \alpha_{t,m,k} c_k, \qquad \sum_{k \in \mathrm{TopS}(r_{t,m})} \alpha_{t,m,k} = 1. \tag{4}$$

The codebook is shared across token positions and trajectories. Multiple tokens and weighted combinations provide a compositional representation within this shared space. When $S = 1$, each token reduces to a single codebook entry.

We mainly use action reconstruction to learn the latent tokens. The action decoder produces $\tilde{\mathbf{a}}_t = g_\omega(z_t)$, a reconstruction of the demonstrated chunk. We train the LAM with

$$\mathcal{L}_{\text{LAM}} = \lambda_{\text{rec}}\mathcal{L}_1(\tilde{\mathbf{a}}_t, \mathbf{a}_t) + \lambda_{\text{commit}}\mathcal{L}_{\text{commit}} + \lambda_{\text{code}}\mathcal{L}_{\text{codebook}}. \tag{5}$$

The reconstruction loss penalizes differences between the decoded and demonstrated actions. The commitment loss penalizes the distance between the continuous encoder output and its codebook representation, while codebook regularization encourages use of different entries to prevent collapse. Together, these objectives learn a compact representation whose content is tied to the action chunk. We then freeze the posterior encoder, action decoder, and codebook to establish a common movement space for the policy and prior.

### 3.2 Training a Latent-Conditioned WAM with Latent Override

The latent-conditioned policy $\pi_\theta$ learns to generate actions from the current observation and a movement predicted in the latent action space. We instantiate $\pi_\theta$ with Cosmos-Policy (Kim et al., 2026), a world–action model (WAM). For each demonstration, the frozen LAM posterior extracts $z_t$ using Eq. 2. We replace the policy's language condition with these latent tokens and train it to predict $\mathbf{a}_t$ from $(o_t, z_t)$, retaining its action representation, action decoder, and training objective on ordinary demonstrations. The LAM decoder supervises the movement representation, while the policy provides the observation-conditioned action generator used for execution.

The WAM must respond to the latent condition for movement selection to control its behavior. In ordinary demonstrations, the observation, latent, and action target are correlated, allowing the WAM to rely on visual cues while paying little attention to $z_t$. This shortcut can undermine adaptation: changing the movement predicted by the prior may have little effect on the generated actions.

We therefore introduce *Latent Override*, a novel training technique encouraging the WAM to use the latent to distinguish alternative movements. For an example $i$, we randomly select a donor chunk $j$ with a similar initial configuration and a different movement. We retain the observation $o_i$, substitute the donor latent $z_j$, and supervise the donor action chunk $\mathbf{a}_j$:

$$(o_i, z_i, \mathbf{a}_i) \quad \longrightarrow \quad (o_i, z_j, \mathbf{a}_j). \tag{6}$$

Matching the initial configuration makes the donor movement a more appropriate target for the retained observation. We apply the standard WAM action loss to overridden examples and mix them with ordinary demonstrations. The ordinary examples retain supervision for execution in the observed scene; the overridden examples pair a retained observation with an alternative movement and action target, encouraging the WAM to resolve the target through the latent condition. The LAM and codebook remain fixed throughout this stage.

### 3.3 Latent Prior and Few-Shot Adaptation

The latent prior predicts an atomic movement using information available at deployment. Since the LAM posterior requires future observations and demonstrated actions, the prior learns to predict its representation from the short observation history $\mathbf{h}_t$ and instruction $\ell$. It encodes these inputs, appends $M$ learned latent queries, and predicts codebook scores $\hat{r}_{t,m}$ for each query. Applying the same sparse composition rule as the LAM gives

$$\hat{z}_{t,m} = \sum_{k \in \text{TopS}(\hat{r}_{t,m})} \hat{\alpha}_{t,m,k} c_k, \qquad \hat{z}_t = (\hat{z}_{t,1}, \ldots, \hat{z}_{t,M}), \tag{7}$$

where $\hat{\alpha}_{t,m,k}$ are nonnegative weights normalized over the selected entries. Sharing the frozen codebook ensures that the prior produces representations in the same space used to condition the WAM. Our implementation uses Qwen3.5-0.8B (Bai et al., 2025) as the prior backbone.

The prior learns from both posterior code targets and demonstrated actions. Let $y_t^q$ denote the code targets supplied by the frozen posterior and $\hat{r}_t = (\hat{r}_{t,1}, \ldots, \hat{r}_{t,M})$ the prior's codebook logits. We train the prior with

$$\mathcal{L}_{\text{prior}} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}}(\hat{r}_t, y_t^q) + \lambda_{\text{rec}}\mathcal{L}_1\left(g_\omega(\hat{z}_t), \mathbf{a}_t\right). \tag{8}$$

The code-classification loss aligns the prior's predictions with the posterior targets and the reconstruction term passes the predicted latent through the frozen LAM decoder and penalizes error relative to the demonstrated actions. This term supervises the movement decoded from the predicted representation in addition to its agreement with the posterior codes. The decoder parameters remain fixed while this loss trains the prior.

Few-shot adaptation involves only parameter updates in the prior adapter. Given demonstrations of a new task, we keep the LAM, codebook, and WAM frozen and update only this adapter module, as in our experimental protocol. Adaptation learns how new instructions and observations map into the established movement space while reusing the WAM's action generator. At deployment, the adapted prior predicts $\hat{z}_t$ and the WAM generates an action chunk conditioned on $(o_t, \hat{z}_t)$. Movement selection and execution repeat in closed loop as new observations arrive.

# 4 EXPERIMENTS

We evaluate whether the atomic-movement interface supports few-shot adaptation while retaining effective low-level control. Experiments on a Franka Panda arm cover ten base tasks and six adaptation tasks. We first assess base-task success, then examine adaptation performance and cost, and finally analyze the learned representation and influence of several key design factors.

There are three major findings. **(1) Base-task performance:** our method achieves 65.5% average success, improving over Cosmos-Policy by 11.0 percentage points and performing comparably to $\pi_{0.5}$ at 66.0% (Table 1). **(2) Few-shot adaptation:** with 6–8 demonstrations per task, updating only the prior adapter achieves 58.33% average success, compared with 50.0% and 55.0% for full-model fine-tuning of Cosmos-Policy and $\pi_{0.5}$, respectively (Table 2). **(3) Adaptation cost:** our method takes 97 minutes on one GPU, compared with 200 and 124 minutes on four GPUs for the respective baselines (Table 3). Together, these results support reusing the learned movement space and WAM for adaptation from a few demonstrations.

## 4.1 EXPERIMENTS SETUP

**Real-world hardware setup.** We deploy all of our experiments on a 7-DoF Franka Panda arm. For all policies evaluated, we use a front-side RealSense D435 and a wrist-mounted D405 (Keselman et al., 2017) for localized observation. We collect all of our demonstrations through leader-follower teleoperation using two identical Panda arms, ensuring high-fidelity stable trajectories.

**Tasks design.** To evaluate the generalization ability of our framework, we collect as many tasks as possible while limiting the number of trajectories per task. Specifically, we collect 10 real-world tasks, each with 20 demonstrations, resulting in 200 demonstrations in total. Of these, 180 trajectories are used for training, while the remaining 20 are used for evaluation. The tasks encompass diverse motion trajectories and are organized into four major categories: ***(i)*** Cup Stacking, which involves stacking different cups into a target cup; ***(ii)*** Pick and Place, which involves interacting with various objects and placing them at target locations; ***(iii)*** Object Pushing, which involves using the robot gripper to push objects to target locations; and ***(iv)*** Tool Usage, which involves using a tool to complete a task. Overall, the tasks are designed to cover a broad range of scenarios for evaluating the generalization ability of the models.

In addition to evaluating accuracy on basic tasks, we further design a few-shot experiment to assess the transferability of different policies. For the few-shot tasks, we consider tasks with distinct setup that nevertheless share some movements with the base tasks, thereby facilitating knowledge transfer. We collect tasks from three major categories: ***(i)*** Novel Objects, which involve goals similar to those of the base tasks but require interaction with previously unseen objects; ***(ii)*** Novel Arrangement Poses, which involve similar goals but take place in entirely unseen arrangement configurations; and ***(iii)*** Long-Horizon Tasks, which consist of several previously seen tasks that must be completed sequentially. We collect approximately 6–8 demonstrations for each task, which are used to adapt the base policies. A complete description of the base and few-shot tasks is provided in the supplementary section B.

**Implementation briefing.** We adopt the WAM training of Cosmos-Policy (Kim et al., 2026), initialized with a Cosmos-Predict 2.5 (Ali et al., 2025) checkpoint. Since the action solution space

Table 1: Task success rates (%) on the base training tasks.

| Method | Cup Stacking | | Pick and Place | | | Object Pushing | | | Tool Usage | | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | |
| Cosmos-Policy Baseline | 45 | 55 | 80 | 25 | 30 | 55 | 65 | 85 | 40 | 65 | 54.5 |
| $\pi_{0.5}$ | 65 | 55 | 90 | 60 | 55 | 55 | 70 | 95 | 55 | 60 | 66.0 |
| **Ours** | 70 | 65 | 90 | 55 | 65 | 50 | 55 | 95 | 60 | 50 | 65.5 |

Table 2: Task success rates (%) on the few-shot task setting.

| Method | Novel Objects | | Novel Arrangement Poses | | Long-Horizon Tasks | | Avg |
|---|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | T5 | T6 | |
| Cosmos-Policy Baseline | 45 | 75 | 40 | 20 | 65 | 55 | 50.0 |
| $\pi_{0.5}$ | 60 | 60 | 45 | 25 | 70 | 70 | 55.0 |
| **Ours** | 60 | 55 | 60 | 40 | 70 | 65 | **58.33** |

is small, we use Qwen3.5-0.8B (Bai et al., 2025) as our prior model backbone. We compare our method with the Cosmos-Policy baseline and $\pi_{0.5}$ (Intelligence et al., 2025) under identical training settings, using the same base tasks and 20,000 training steps. For the few-shot experiments, our model only requires fine-tuning the lightweight prior adapter, whereas the compared baselines undergo full-model fine-tuning. All few-shot experiments use 500 training steps and a learning rate one-tenth that used for training on the base tasks.

### 4.2 Base Task Performance

Table 1 evaluates performance across ten base tasks with diverse initial object poses. Our method achieves an average success rate of 65.5%, improving over the Cosmos-Policy baseline by 11.0 percentage points and attaining performance comparable to $\pi_{0.5}$ (66.5%). Our method also matches or exceeds $\pi_{0.5}$ on six of the ten tasks. These results establish that introducing a reusable movement interface preserves competitive manipulation capability while providing potential for dealing with out-of-distribution cases.

The improvement over the baseline is consistent with our design for generalization. The atomic-movement interface separates instruction-conditioned movement selection from precise execution: the prior selects combinations of learned movement codes, while the WAM uses the current observation to generate the corresponding actions. Sparse codebook composition encourages reuse of movement structure across demonstrations, reducing the need to learn each instruction–configuration combination independently. Together, these mechanisms encourage the policy to follow the requested movement while remaining responsive to the current scene, achieving better generalization.

### 4.3 Few-shot Adaption Performance

Updating only the prior adapter achieves competitive few-shot success while keeping the policy fixed. With 6–8 demonstrations per task, our method achieves 58.33% average success across novel objects, unseen arrangement poses, and long-horizon tasks, compared with 50.0% for Cosmos-Policy and 55.0% for $\pi_{0.5}$ under full-model fine-tuning (Table 2). The strongest advantage appears in novel arrangement poses, supporting the value of reusing learned movements when familiar task goals must be executed under new spatial configurations. These results show that adaptation through movement selection can retain competitive performance without updating the policy.

Table 3: Resource requirements for few-shot adaptation.

| Method | Time (min) | GPUs | Memory (GB) |
|---|---|---|---|
| Cosmos-Policy | 200 | 4 | 80 |
| $\pi_{0.5}$ | 124 | 4 | 93 |
| **Ours** | **97** | **1** | **8.2** |

During adaptation, the LAM, codebook, and policy remain fixed, while only the prior adapter is updated to learn how new observations and instructions map into this established movement space. Table 3 reports the adaptation resource requirements. Our method completes adaptation in an average of 97 minutes per task on one GPU, compared with 200 minutes for Cosmos-Policy and 124 minutes for $\pi_{0.5}$, both using four GPUs. Training memory required is also substantially lower, at 8.2 GB compared with 80 GB and 93 GB. By limiting parameter updates to the prior adapter, our method avoids maintaining gradients for the policy. These results show that adapting movement selection reduces the measured per-task adaptation cost while retaining competitive task success.

### 4.4 Ablation Study

**Latents PCA.** To examine the structure of the learned latent actions, we fit Principal Component Analysis (PCA) (Jolliffe & Cadima, 2016) on training-set posterior latents and project latent tuples from held-out trajectories. As shown in Figure 3, each action chunk is represented by six 32-dimensional latent tokens, with marker area indicating its usage frequency and color indicating the mean motion statistic of the associated action chunks; the first two principal components retain 73.4% of the total latent variance (62.1% and 11.3%, respectively), indicating that the two-dimensional projection preserves most of the representation's global variation.

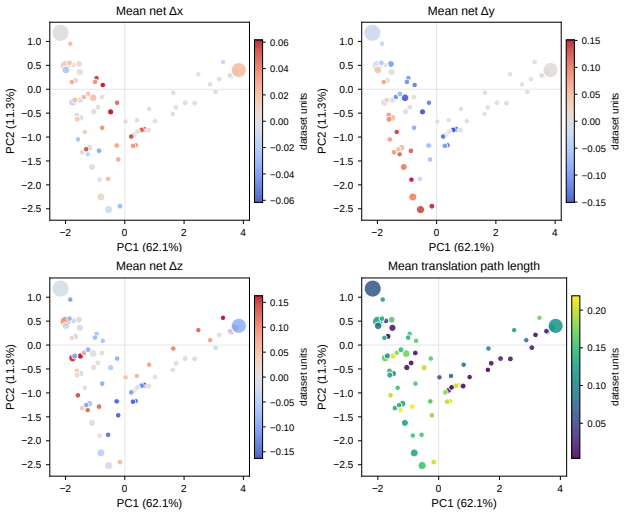



Figure 3: Visualized learned latent-action space.

Despite receiving no explicit displacement or trajectory-length supervision, the learned latent space exhibits consistent kinematic organization: the four major statistics all show a clear separation pattern for high and low values, with $\Delta y$ and translation path length being the most prominent. This structured organization demonstrates that our LAM learns reusable latent codes that clearly capture both the direction and magnitude of robot motion, rather than simply memorizing the motion pattern for each training task individually.

**WAM Latent Override.** To assess agreement between the movement condition and WAM output, we measure the L1 distance between the LAM-decoded trajectory and the WAM-predicted trajectory conditioned on the same latent during real-world execution. Table 4a reports an L1 of 0.1535 without Latent Override and 0.1087 with it. It shows that latent override enables the WAM to follow the conditioning latent action substantially more faithfully, highlighting the importance of this design.

**LAM Token Count.** We vary the number of latent tokens $M$ per action chunk to assess reconstruction quality (Table 4b). Six tokens yield the lowest reconstruction L1 of 0.008, compared with 0.015, 0.016, and 0.013 for two, four, and eight tokens, respectively. We select six tokens for our tested setting, yet the number of token does not affect too much on the LAM performance.

**Prior History Length.** We vary the number of observation-history frames to evaluate their effect on reconstruction from prior-predicted latents (Table 4c). Reconstruction L1 decreases from 0.126 to 0.118 and 0.114 with two, three, and four frames, respectively, indicating that additional history improves reconstruction over the tested range, with a smaller gain from the fourth frame.

### 4.5 Qualitative Analysis

We provide qualitative analysis in Figure 4, showing two chunks from different stages across multiple task categories. As illustrated, the latent prior first predicts an atomic movement, which then

Table 4: Ablations of Latent Override, LAM token count, and prior observation history. Bold indicates the best result in each ablation, which we use as default setting for all the experiments.

**(a)** Latent Override.

| Setting | Action L1 ↓ |
|---|---|
| w/ latent override | **0.1087** |
| w/o latent override | 0.1535 |

**(b)** LAM token count.

| Tokens | Recon. L1 ↓ |
|---|---|
| 2 | 0.015 |
| 4 | 0.016 |
| 6 | **0.008** |
| 8 | 0.013 |

**(c)** Prior history length.

| Frames | Recon. L1 ↓ |
|---|---|
| 2 | 0.126 |
| 3 | 0.118 |
| 4 | **0.114** |

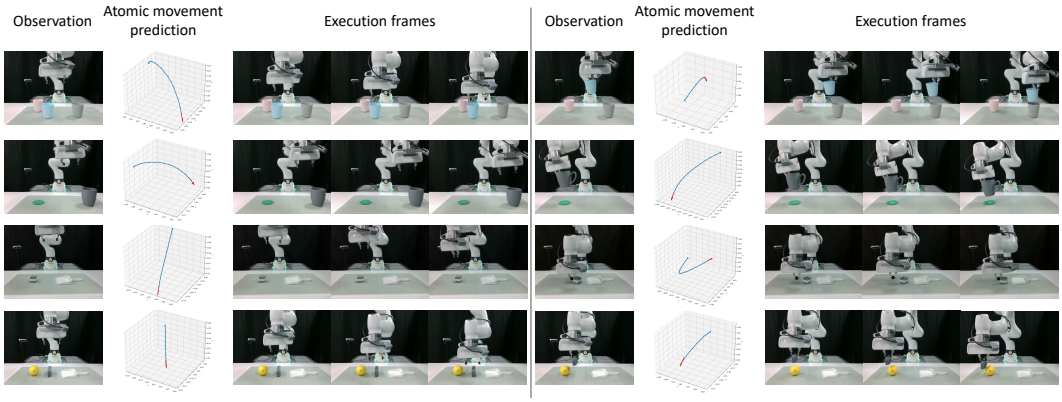



Figure 4: Qualitative examples of robot execution through atomic movement conditioning. We visualize the predicted latent atomic movement by feed them through LAM action decoder to obtain the reconstructed trajectory. The end of trajectory is marked with a red arrow.

conditions the WAM to generate the corresponding action trajectory. The predicted atomic movements capture the intended directions toward the target poses, while the conditioned WAM outputs faithfully follow these directions with more precise pose localization and control. For example, in the second chunk of the third task, the prior predicts an initial slight backward movement to adjust the gripper position. The WAM follows this prediction and reaches the correct pushing pose. These results further support our core motivation: shifting the burden of planning and generalization to atomic-movement prediction allows the WAM to focus on accurate low-level action generation.

## 5 Conclusion

We presented an atomic-movement interface for adapting robot policies through reusable movement representations. A latent prior selects a movement from the instruction and observation history, and a latent-conditioned WAM generates the corresponding actions in the current scene. Sparse codebook composition provides a shared representation, while Latent Override encourages the WAM to follow the selected movement. This separation enables adaptation by updating only the prior adapter. Experiments on ten base tasks and six few-shot tasks show competitive manipulation performance and a higher average adaptation success rate than full-model fine-tuning of the two evaluated baselines, with lower reported adaptation time and GPU requirements. These findings support movement selection as a practical focus for adaptation when learned control capabilities can be reused.

Our evaluation is limited to one robot platform and adaptation tasks that share movement components with the base tasks. Extending the approach to tasks requiring motor skills beyond the learned embodiment remains an open question. Future work will study how broader demonstration data can expand the movement space and evaluate transfer across more diverse tasks and embodiments.

### AI USE STATEMENT

In this work, we used generative AI tools for implementation for parts of the modules in the coding projects, as well as polishing the paper language. We have not used generative AI tools for original paper writting. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We provide the anonymous downloadable source code in the supplementary experiments. The source code will be further sorted up and open-sourced after the paper being accepted.

## REFERENCES

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi_0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15391*, 2024. URL https://arxiv.org/abs/2402.15391.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026. URL https://arxiv.org/abs/2606.15768.

Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, Jianyu Chen, and Jiang Bian. villa-X: Enhancing latent action modeling in vision-language-action models. *arXiv preprint arXiv:2507.23682*, 2025. URL https://arxiv.org/abs/2507.23682.

Yi Chen, Yuying Ge, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for robot manipulation. *arXiv preprint arXiv:2412.04445*, 2024. URL https://arxiv.org/abs/2412.04445.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. *arXiv preprint arXiv:2503.18938*, 2025. URL https://arxiv.org/abs/2503.18938.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. Robotic task generalization via hindsight trajectory sketches. In *International Conference on Learning Representations*, 2024. URL https://rt-trajectory.github.io/pdf/RT_Trajectory.pdf.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024. URL https://arxiv.org/abs/2409.01652.

Dongyoon Hwang, Byungkun Lee, Dongjin Kim, Hyojin Jang, Hoiyeong Jin, Jueun Mun, Minho Park, Hojoon Lee, Hyunseung Kim, and Jaegul Choo. 3D HAMSTER: Bridging planning and control in hierarchical vision language action models through 3D trajectory guidance. *arXiv preprint arXiv:2606.31329*, 2026. URL https://arxiv.org/abs/2606.31329.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Yuxin Jiang, Yuchao Gu, Ivor W. Tsang, and Mike Zheng Shou. Olaf-World: Orienting latent actions for video world modeling. *arXiv preprint arXiv:2602.10104*, 2026. URL https://arxiv.org/abs/2602.10104.

Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374(2065):20150202, 2016.

Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1–10, 2017.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. *arXiv preprint arXiv:2502.05485*, 2025. URL https://arxiv.org/abs/2502.05485.

Yihan Lin, Jiawei He, Shifeng Bao, Chen Zhao, Yang Li, Xiaobo Wang, Yan Wang, Cheng Chi, and Jing Zhang. JEPA-WAM: Learning vision-language-action policies with joint-embedding world modeling. *arXiv preprint arXiv:2608.09381*, 2026. URL https://arxiv.org/abs/2608.09381.

Fangchen Liu, Kuan Fang, Pieter Abbeel, and Sergey Levine. MOKA: Open-world robotic manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024. URL https://arxiv.org/abs/2403.03174.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. QueST: Self-supervised skill abstractions for learning continuous control. *arXiv preprint arXiv:2407.15840*, 2024. URL https://arxiv.org/abs/2407.15840.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Yifu Qiu, Zheng Zhao, Waylon Li, Yftah Ziser, Anna Korhonen, Shay B. Cohen, and Edoardo M. Ponti. Self-improving world modelling with latent actions. *arXiv preprint arXiv:2602.06130*, 2026. URL https://arxiv.org/abs/2602.06130.

Hongyu Qu, Jianzhe Gao, Xiaobin Hu, Shaohuan Yang, Xinlei Yu, Rui Yan, Wenguan Wang, Xiangbo Shu, and Shuicheng Yan. Dual latent memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2607.07608*, 2026. URL https://arxiv.org/abs/2607.07608.

Xiaofan Wang, Xingyu Gao, Jianlong Fu, Zuolei Li, Dean Fortier, Galen Mullins, Andrey Kolobov, and Baining Guo. LoLA: Long horizon latent action learning for general robot manipulation. *arXiv preprint arXiv:2512.20166*, 2025. URL https://arxiv.org/abs/2512.20166.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024. URL https://arxiv.org/abs/2410.11758.

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

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024. URL https://arxiv.org/abs/2412.10345.

# A Additional Implementation Details

## A.1 Observation and Action Representations

Each training example contains a chunk of $H = 16$ actions and $H + 1 = 17$ synchronized observations from two RGB cameras. Each camera image is resized to $224 \times 224$. An action has seven components: three Cartesian end-effector displacements, three relative Euler-angle components, and one binary gripper command. The preprocessing code computes translation as $p_{t+1} - p_t$ and rotation from $R_{t+1}R_t^{-1}$, expressed in XYZ Euler angles. Translation is measured in metres and rotation in radians before normalization. The gripper command is $-1$ for open and $+1$ for closed, using a physical width threshold of $0.07\,\mathrm{m}$ during preprocessing.

## A.2 LAM Architecture and Sparse Code Selection

The LAM posterior combines video, action, and instruction tokens before extracting six latent queries. A frozen Wan2.1 video tokenizer maps the 17-frame clip from each camera to a latent video with 16 channels and spatial resolution $28 \times 28$. These latents are divided into $4 \times 4$ patches and projected to width 128. Learned time, view, patch, and modality embeddings identify the tokens; a two-layer spatiotemporal encoder processes the video tokens. The posterior then concatenates video tokens, projected action tokens, instruction tokens, and $M = 6$ learned queries and applies a three-layer Transformer. Each query output is projected to $d = 32$ dimensions. For string instructions, the LAM uses a learned 4,096-entry embedding table indexed by deterministic hashes of up to 16 whitespace-separated words; the prior uses the Qwen tokenizer instead.

The codebook contains $K = 32$ entries and computes top-$S$ weights with $S = 2$ and temperature $\tau = 0.2$. Given logits $r_{m,k}$ and their selected support $\mathcal{S}_m = \mathrm{TopS}(r_m)$, the soft weights are

$$w_{m,k} = \frac{\mathbb{I}[k \in \mathcal{S}_m] \exp(r_{m,k}/\tau)}{\sum_{j \in \mathcal{S}_m} \exp(r_{m,j}/\tau)}. \tag{9}$$

The saved configuration enables straight-through selection. Writing $h_m$ for the one-hot vector at $\arg\max_k w_{m,k}$ and sg for stop-gradient, the implemented token is

$$z_m = \sum_k [w_{m,k} + \mathrm{sg}(h_{m,k} - w_{m,k})]\, c_k. \tag{10}$$

Thus, the forward pass uses one codebook entry per token, while gradients through the selection weights follow the top-2 soft distribution. Disabling straight-through selection produces the weighted mixture in Eq. 1. This distinction applies to both posterior and prior tokens.

The action decoder receives only the latent tuple. It projects the six latent tokens to width 128, processes them with a separate three-layer Transformer, flattens the resulting token features, and maps them to the $16 \times 7$ action chunk. It does not receive the initial image or instruction in this configuration. Consequently, action reconstruction requires movement information to pass through the codebook interface. The implementation also contains a video reconstruction branch, but its loss weight is zero in the inspected LAM configuration.

The LAM is trained with action reconstruction and codebook regularization:

$$\mathcal{L}_{\mathrm{LAM}} = 10\mathcal{L}_{\mathrm{act}} + 0.1\mathcal{L}_{\mathrm{commit}} + 0.03\mathcal{L}_{\mathrm{sparse}} + 0.03\mathcal{L}_{\mathrm{usage}}. \tag{11}$$

Here, $\mathcal{L}_{\mathrm{act}}$ is mean Smooth L1 loss with transition parameter 1. The commitment term is the mean squared error between $z$ and the detached continuous encoder output $\mathrm{sg}(z^e)$. For soft code weights $w$, the sparsity term averages their per-token entropy, while the usage term is the negative entropy of their batch-averaged distribution. These terms encourage concentrated assignments and diverse code usage, respectively. No video reconstruction loss is used.

### A.3 Prior Architecture and Training Objective

The prior predicts the movement representation from the instruction and four observation frames, including the current frame. At each timestep, the two camera views are processed by Qwen3.5-0.8B through its native visual input path. Six learned queries are appended to the input, and a linear classification head maps their features to codebook logits. The predicted latent $\hat{z}$ uses the same top-2 and straight-through selection rules as the LAM (Eqs. 9–10).

The frozen LAM posterior supervises both the distribution of code weights and the highest-weight code. Let $w^q_m$ denote its soft weights for token $m$, $y^q_m = \arg\max_k w^q_{m,k}$ the corresponding hard target, and $p_m$ the prior's full 32-way softmax distribution. We define

$$\mathcal{L}_{\mathrm{soft}} = -\mathbb{E}\left[\sum_{k=1}^{K} w^q_{m,k} \log p_{m,k}\right], \qquad \mathcal{L}_{\mathrm{hard}} = -\mathbb{E}\left[\log p_{m,y^q_m}\right], \tag{12}$$

where the expectations average over training examples and latent-token positions.

Action reconstruction additionally supervises the movement decoded from the predicted latent. The base prior objective is

$$\mathcal{L}_{\mathrm{prior}} = \mathcal{L}_{\mathrm{soft}} + 0.1\mathcal{L}_{\mathrm{hard}} + 2\mathcal{L}_{\mathrm{act}}\big(g_\omega(\hat{z}), \bar{\mathbf{a}}\big). \tag{13}$$

Here, $\bar{\mathbf{a}}$ is the normalized demonstration action chunk and $\mathcal{L}_{\mathrm{act}}$ is the Smooth L1 loss defined above. The LAM decoder and codebook entries remain fixed, while reconstruction gradients pass through the decoder to update the prior.

### A.4 WAM Conditioning and Latent Override

The WAM receives the latent tuple through a dedicated input frame in its latent video representation. The six 32-dimensional tokens are tiled into the otherwise blank first conditioning frame using the same packing routine used for action chunks. The following conditioning frames contain current proprioception, the wrist image, and the external-camera image. The action frame remains a prediction target. T5 embeddings are zeroed and text-conditioning dropout is set to one, so language reaches the controller through the prior-predicted movement rather than through a separate text condition. The movement frame is excluded from the reconstruction loss. During WAM training, the posterior is evaluated without gradients; this online encoding path omits the optional LAM language argument and uses its null-language embedding.

Latent Override selects donors by robot pose and physical gripper state. For recipient $i$, candidate donors must have the same open/closed gripper state and, when the configured cross-episode constraint is enabled, come from a different trajectory. A random candidate pool is drawn without replacement. Using normalized initial position and Euler-angle components $s_{i,1:6}$, the selected donor minimizes

$$d(i,j) = \sqrt{\frac{1}{6}\sum_{r=1}^{6}(s_{i,r} - s_{j,r})^2}. \tag{14}$$

Table 5: Instructions for the ten base tasks, grouped by manipulation family.

| ID | Family | Instruction |
|---|---|---|
| T1 | Cup stacking | Stack the blue cup into the grey cup. |
| T2 | | Stack the pink cup into the grey cup. |
| T3 | Pick and place | Pick up the grey cup and place it on the green plate. |
| T4 | | Pick up the orange cube and place it in the bowl. |
| T5 | | Pick up the white cube and place it in the bowl. |
| T6 | Object pushing | Push the grey cube into the dustpan. |
| T7 | | Push the orange cube into the dustpan. |
| T8 | | Push the white cube into the dustpan. |
| T9 | Tool usage | Sweep the grey cube into the dustpan. |
| T10 | | Sweep the orange cube into the dustpan. |

The donor is accepted only if this distance is below a threshold; otherwise, the original example is retained. The distance is dimensionless because it uses normalized coordinates. The implementation does not require a different task label or explicitly test for a different movement. It also does not compare object poses or contact states.

For an accepted donor, the recipient's current visual and proprioceptive observations are retained, while the action target and the video–action inputs used to obtain the posterior latent are replaced by the donor's. The override flag restricts supervision to the action prediction and multiplies its loss by a coefficient $\eta$. Future-image, future-proprioception, and value targets are excluded for these examples because they are not established outcomes of the substituted movement in the recipient scene. Ordinary examples retain the inherited Cosmos-Policy objective.

The WAM objective is a masked, noise-weighted squared denoising error. If $X$ is the clean packed latent tensor, $\hat{X}_\theta$ its denoised prediction, $u$ indexes latent tensor elements, and $m_{i,u}$ is the loss mask, the implemented reduction has the form

$$\mathcal{L}_{\mathrm{WAM}} = \frac{1}{B|\mathcal{U}|} \sum_{i,u} m_{i,u}\, w(\sigma_{i,u}) \big(\hat{X}_{\theta,i,u} - X_{i,u}\big)^2. \tag{15}$$

Here, $w$ is the inherited EDM noise weighting. For override examples, the mask is zero outside the action frame and includes the factor $\eta$ inside it. The final mean is over the full tensor, rather than being renormalized by the number of unmasked elements. The attempted override fraction and the accepted fraction can differ because donor matching can fail.

## B Experimental Setting

We use the Franka Panda setup described in Section 4. The base dataset contains 20 demonstrations for each of ten tasks, with 180 trajectories for training and 20 held out for evaluation. Each few-shot task provides approximately 6–8 demonstrations.

### B.1 Task Descriptions

The base tasks cover cup stacking, pick-and-place, object pushing, and tool use (Table 5). Stacking requires nesting a source cup in a target cup; pick-and-place requires lifting and transporting an object. Pushing moves a cube into a dustpan through direct gripper contact, whereas sweeping uses a tool.

The six adaptation tasks are listed in Table 6. Novel-object tasks introduce a yellow cup and a lemon while retaining stacking and sweeping goals. Novel-arrangement tasks change the spatial configurations for blue-cup stacking and orange-cube pick-and-place. Long-horizon tasks require manipulating two objects into a common target, testing successive movement selection with the same frozen WAM.

Table 6: The six few-shot tasks and their distribution shifts relative to base training.

| ID | Setting | Instruction |
|---|---|---|
| T1 | Novel objects | Sweep the lemon into the dustpan. |
| T2 | | Stack the yellow cup into the green cup. |
| T3 | Novel arrangements | Stack the blue cup into the grey cup. |
| T4 | | Pick up the orange cube and place it in the bowl. |
| T5 | Long horizon | Stack the blue and pink cup into the grey cup. |
| T6 | | Pick up the white and orange cube and place it in the bowl. |

Table 7: Principal optimization hyperparameters. Prior adaptation uses an effective batch size of 16 through gradient accumulation. WAM batch size is reported per GPU. Values are drawn from the available training configurations; final run lengths remain to be confirmed.

| Hyperparameter | LAM | WAM | Prior | Prior adaptation |
|---|---|---|---|---|
| Learning rate | $2 \times 10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-5}$ |
| Batch size | 16 | 30/GPU | 16 | 16 |
| Weight decay | $5 \times 10^{-4}$ | 0.1 | $10^{-5}$ | $10^{-5}$ |
| Warmup updates | 300 | 1,000 | 100 | 25 |

### B.2 Evaluation Protocol

Task success is averaged equally across tasks. Reconstruction L1 measures mean absolute error between decoded and demonstrated actions; latent-following L1 compares the LAM-decoded and WAM-predicted actions under the same latent. These evaluation metrics differ from the Smooth L1 training objective.

For the latent visualization, each $6 \times 32$ tuple is flattened into 192 dimensions. PCA is fitted on 4,096 training windows and applied to 1,361 held-out windows; the first two components explain 73.35% of the variance. Colors indicate action-derived movement statistics.

## C Hyperparameters and Training Procedure

The LAM posterior and action decoder each use three Transformer layers, hidden dimension 128, four attention heads, and dropout 0.15. The prior uses rank-8 LoRA with scaling parameter 16 during base training, and the WAM follows the 2B Cosmos-Policy architecture. Table 7 lists the principal optimization settings. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$ for the LAM and prior and $(0.9, 0.99)$ for the WAM. The LAM and prior use linear warmup followed by cosine learning-rate decay.

Few-shot adaptation updates the prior's learned queries, positional/type embeddings, and classification head while freezing the LAM, codebook, WAM, and Qwen parameters, including LoRA. If state conditioning is enabled, its encoder is also updated at learning rate $10^{-4}$. Four micro-batches of size four give an effective batch size of 16.

The default Latent Override fine-tuning stage uses learning rate $10^{-5}$, an attempted override fraction of 10%, and override loss weight $\eta = 0.1$. Donor matching considers at most 512 candidates and accepts a normalized pose distance of at most 0.1.

**Inference.** The prior predicts a latent tuple from four observation frames, and the WAM generates a 16-step action chunk. The deployment configuration uses five denoising steps and a nominal control rate of 10 Hz. The LAM decoder is used only for optional movement visualization.