TRANSFERABLE BEHAVIOR EMBEDDINGS FROM UNLABELED VIDEOS FOR POLICY LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A robot policy chooses better actions when it knows how the scene is changing, where that change leads, and how far the current step has progressed. We learn this information from unlabeled manipulation videos as behavior embeddings that describe subprocesses: coherent video segments discovered from the embeddings at several granularities. Each embedding combines a motion channel describing ongoing motion with a scene configuration channel describing the configuration a subprocess reaches and its relation to the current scene. Three tasks give these channels their meaning: next-frame prediction, endpoint reconstruction, and status scoring. Learning requires no action, language, subtask, or progress annotations. The frozen encoder then computes embeddings for policy demonstrations. Predicting these embeddings from the policy's action features regularizes those features with no additional inference cost. Experiments show consistent improvements in task success across vision–language–action and world-action models. On a real robot, behavior supervision also improves task success and generalization to unseen instruction combinations and appearance changes. Probes also show that motion, reached configuration, and execution status are readable from the embeddings.

1 INTRODUCTION

Robot manipulation policies often struggle to generalize beyond the instructions and visual conditions in their demonstrations. For example, a policy fine-tuned to put a cup in a drawer and a cube in a container may, when asked to put the cube in the drawer, open the drawer but place the cube in the familiar container (Fig. 1). Plausible motions alone are insufficient: the policy must track how the scene changes, where those changes should lead, and how far the current step has progressed. Demonstrations pair images with actions but do not explicitly label these relations. Action supervision can therefore reward features tied to familiar appearances and object–destination pairings, leaving the policy vulnerable to new pairings or even a different tablecloth.

Existing approaches address parts of this gap: vision–language–action models provide semantic priors for interpreting instructions (Black et al., 2024; Intelligence et al., 2025), world-action models learn how observations evolve (Yuan et al., 2026), and pretrained visual encoders capture appearance and layout (Oquab et al., 2023; Tschannen et al., 2025). However, they do not explicitly supervise the joint relation between ongoing motion, its outcome, and execution status: where a step ends and whether it has ended. Neither language nor a single frame fully specifies this relation, and annotating subtask boundaries or progress at scale is costly.

Our observation is that the missing knowledge is available in hindsight: watching a stretch of video to its end reveals what was being done, where it led, and when it was done, without any label, and manipulation videos exist in large numbers (Khazatsky et al., 2024). We therefore learn *behavior embeddings* from unlabeled videos: an encoder reads a whole window of frames and describes each transition by the *subprocess* it belongs to, a stretch in which the scene changes coherently, through what is being done in it, what the scene looks like when it ends, and whether it has ended.

The policy cannot access future observations at inference, but can learn to predict what hindsight reveals. We use the frozen encoder to compute embeddings offline for the policy's demonstrations, then train a small auxiliary head to predict them from the features used for action prediction. This loss encourages features of the current image and instruction to capture the manipulated object, its destination, and execution status, reducing reliance on familiar appearances and pairings. The

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.

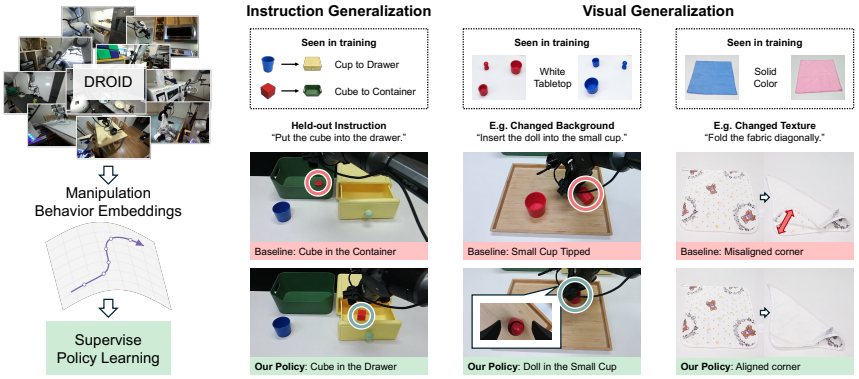


Figure 1: **Learning behavior supervision for generalization.** Behavior embeddings learned from DROID videos supervise policy training (left). On Mobile ALOHA, the supervised policy generalizes to an unseen object–destination pairing (middle) and to a changed background and cloth texture (right), where the base policy fails.

encoder and head are removed at inference, leaving the policy's inputs, architecture, and inference cost unchanged.

We train one encoder on DROID and use it, frozen, for π_0, $\pi_{0.5}$ and FastWAM on RoboTwin2.0 (Chen et al., 2025c), and for $\pi_{0.5}$ on the real Mobile ALOHA. Behavior supervision improves success on simulation tasks and keeps its advantage under scene randomization; on the real robot it succeeds on held-out instruction combinations where the base policy never does, and it achieves the best performance across the tested background and appearance changes. Probes show that motion, reached configuration, and execution status are decodable, while ablations reveal the contributions of endpoint and status supervision. In summary, we contribute a self-supervised behavior embedding that ties the motion of a subprocess to its reached configuration and status, learned from unlabeled video at several granularities; its use as a regularizing target that improves policy success across policy families at no inference cost; and an analysis of which facts the embedding carries.

2 Related Work

Manipulation representation learning. Manipulation representations differ in their supervision. Language grounds visual semantics (Tschannen et al., 2025; Nair et al., 2022; Karamcheti et al., 2023) and transitions (Ma et al., 2023; Li et al., 2024; Bu et al., 2025); annotated or generated stages (Liu et al., 2026b; Feng et al., 2026) and behavior labels (Hu et al., 2026) supervise dynamics and progress. Recorded states and actions support manipulation-centric pretraining (Jiang et al., 2025), behavior abstractions (Zhou et al., 2025; Mete et al., 2024), joint endpoint, state-membership and action prediction (Liu et al., 2024), and visual–proprioceptive discovery of coherent segments with endpoint prediction and policy supervision (Liu et al., 2025). Video-only methods learn latent actions and skills (Ye et al., 2025; Xu et al., 2023), including variable-duration (Kim et al., 2026a) and optical-flow skills (Fang et al., 2025), align phases across videos (Dwibedi et al., 2019), or measure progress toward goals or reference videos (Zakka et al., 2022; Ma et al., 2022; Ayalew et al., 2025; Yang et al., 2026). Video reconstruction and prediction yield spatiotemporal features (Tong et al., 2022; Wang et al., 2026; Assran et al., 2025; Mur-Labadia et al., 2026), while image reconstruction captures spatial structure (He et al., 2022; Chen et al., 2025a;b). We jointly encode ongoing motion, reached configuration and subprocess status in each transition embedding, using subprocesses discovered from unlabeled video alone at multiple temporal granularities.

Policy learning and generalization. Benchmarks test visual and instruction shifts (Chen et al., 2025c; Fei et al., 2026), where new goal combinations demand different behavior rather than reworded commands. Vision–language–action models transfer semantic priors to control (Black et al., 2024; Intelligence et al., 2025), and predictive policies learn future observations or latent transitions (Li et al., 2026b; Zhang et al., 2026; Kim et al., 2026b; NVIDIA et al., 2026; Yuan et al., 2026; Lin et al., 2026; Sun et al., 2026; Yan et al., 2026). Language alignment, instruction diversity and grounded decomposition support new goal compositions (Myers et al., 2025; Dalal et al., 2026; Peng et al., 2026; Ma et al., 2026; Luo & Wang, 2026). Another route supervises policy features with

auxiliary targets: future visual features (Zheng et al., 2025; Zhao et al., 2026), action-grounded motor embeddings (Xu et al., 2026), jointly learned latent transitions (Liu et al., 2026a), geometry and motion (Li et al., 2026a; Xie et al., 2026), or pretrained world-model features (Dao et al., 2026). We take this route and use behavior embeddings learned from unlabeled video as complementary targets predicted from action features alongside each policy's native objective. Learned without language supervision, they improve visual and instruction generalization while preserving policy inputs and inference cost (Sec. 4).

3 METHOD

We use supervision learned from manipulation videos to improve policies fine-tuned on demonstrations. Consider placing an apple in a pot and then putting the lid on. To choose next actions, the policy needs three facts: how the scene is changing (the gripper approaching the apple or withdrawing), where that change leads (the apple carried to the pot or elsewhere), and how far it has come (on the way to the pot the gripper keeps moving; at the pot it releases).

We learn these facts from videos as a *behavior embedding* z_t, one vector for each transition t between consecutive frames. It describes the *subprocesses* the transition belongs to: segments of video in which the scene changes coherently, such as carrying the apple toward the pot, whose last frames are their *endpoints*. Subprocesses are discovered from the embeddings themselves, without annotations, at a granularity set by a *tolerance* ϵ: strict tolerances favor short coherent segments, while permissive tolerances allow longer subprocesses (Fig. 2, bottom left). Training across granularities exposes the embeddings to subprocess endpoints at multiple horizons. z_t has two parts (Sec. 3.1). The *motion channel* c_t describes the motion under way, e.g., the apple moving in the direction toward the pot, and varies within a subprocess by no more than the tolerance. The *scene configuration channel* s_t describes the scene context of the transition: the configuration the subprocess reaches, e.g., the apple inside the pot, and how the current scene relates to it. Three tasks give the channels these meanings (Sec. 3.2): next-frame prediction, endpoint reconstruction, and status scoring, which separates ongoing frames from the endpoint and orders them, so that the embedding supports judging how far the subprocess has come.

Training has two stages (Fig. 2). Stage 1 (Secs. 3.1–3.2) trains the trajectory encoder $\mathcal{E}$ on videos alone, without actions, language or annotations; because endpoints lie in the future, $\mathcal{E}$ reads a whole window of frames, including those after t. Stage 2 freezes $\mathcal{E}$, computes embeddings offline for policy demonstrations, and trains an auxiliary head to predict them from the policy's action features (Sec. 3.3). This supervision encourages the policy's features to capture the handled object, its destination and execution status from its available inputs, with the aim of improving transfer to new scenes and instructions. The encoder and head are removed at test time.

3.1 BEHAVIOR EMBEDDINGS AND SUBPROCESSES

Windows, grids and the encoder. The encoder works on *windows* of T frames sampled at a fixed stride (Appendix A.1.1), with $T-1$ *transitions*, transition t from frame t to $t+1$. Each frame $\mathbf{o}_t$ is first converted by a pretrained, frozen image encoder Φ_{DC} into a grid $x_t = \Phi_{\mathrm{DC}}(\mathbf{o}_t) \in \mathbb{R}^{H_x \times W_x \times d_x}$ of $H_x \times W_x$ spatial cells with d_x features each, which preserves the layout of the scene (Appendix A.1.1). The bidirectional Transformer trajectory encoder $\mathcal{E}$ with parameters $\Theta_{\mathcal{E}}$ reads all grids in a window and returns one embedding per transition, $(z_1, \ldots, z_{T-1}) = \mathcal{E}(x_1, \ldots, x_T; \Theta_{\mathcal{E}})$, where $z_t \in \mathbb{R}^{d_z}$ may depend on frames after t—harmless because $\mathcal{E}$ is applied only to recorded videos (Sec. 3.3).

Two channels. Each embedding is the concatenation $z_t = [s_t; c_t]$ of the scene configuration channel $s_t \in \mathbb{R}^{d_z - d_c}$ and the motion channel $c_t \in \mathbb{R}^{d_c}$ (widths in Appendix A.1.1; d_c study in Appendix A.2). Subprocess discovery needs c_t to remain coherent within the selected tolerance while one motion continues and to change when another begins. The channel s_t supplements the description of configurations that subprocesses lead to and their relation to the current scene. Both channels have unit length: c_t so that it can be compared by angle, s_t so that its scale stays fixed during embedding learning and policy supervision. Both roles follow from how the channels enter the tasks of Sec. 3.2.

Subprocesses. We compare motion channels by their angle, $\mathrm{dist}(c_p, c_q) = \pi^{-1} \arccos\langle c_p, c_q \rangle \in [0, 1]$. A run of consecutive transitions $[a, b) = \{a, \ldots, b-1\}$ is *coherent* at tolerance $\epsilon \in [0, 1]$ if its diameter $\Delta(a, b) = \max_{a \le p, q < b} \mathrm{dist}(c_p, c_q)$ is at most ϵ; bounding all pairs rather than only

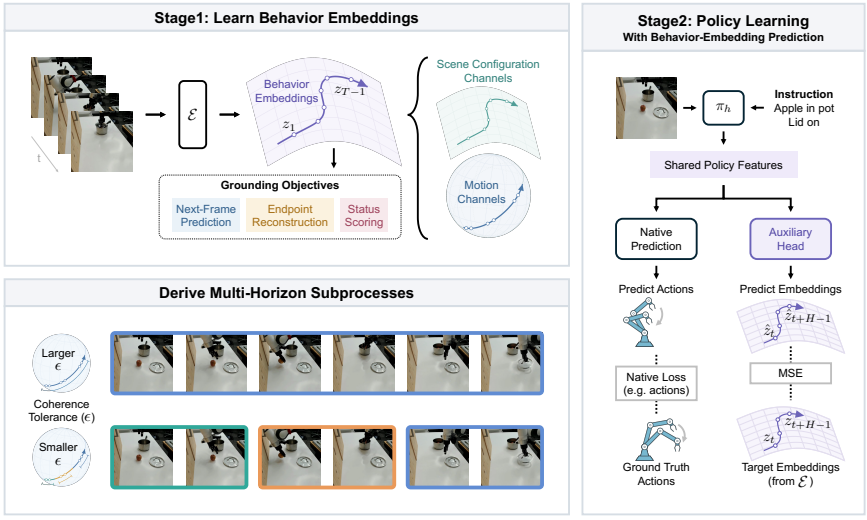


Figure 2: **Overview. Top left:** the trajectory encoder $\mathcal{E}$ maps a window of frames to one behavior embedding per transition; three prediction tasks shape these embeddings. **Bottom left:** transitions whose motion channels agree within a tolerance ϵ form a subprocess; here a strict tolerance gives three subprocesses and a permissive one gives one. **Right:** the frozen $\mathcal{E}$ computes the targets offline; an auxiliary head predicts them from the policy's features π_h; both are removed at test time.

neighbors prevents a run from drifting. Subprocesses are the maximal coherent runs found greedily:

$$g_1 = 1, \quad g_{k+1} = \max\{b \in \{g_k + 1, \dots, T\} : \Delta(g_k, b) \le \epsilon\}, \tag{1}$$

so that the k-th subprocess is $I_k = [g_k, g_{k+1})$. For a transition t, k indexes its subprocess and g its endpoint frame (g_{k+1}); both depend on ϵ (omitted), so another tolerance can assign the same transition a different subprocess and endpoint. Splitting a video where motion loses coherence mirrors how people segment activity into events at several time scales (Zacks et al., 2001; Reynolds et al., 2007, see also Liu et al., 2025). At every training step, each subprocess receives a shared motion *prototype* $\bar{c}_k$, a random positive mixture of its members' motion channels, normalized to unit length: $\bar{c}_k = \mathrm{normalize}_2\left(\sum_{j \in I_k} r_j c_j\right)$ with independent $r_j \sim \mathrm{Unif}(0, 1)$ and $\mathrm{normalize}_2(v) = v/\|v\|_2$. In the tasks (Sec. 3.2), the prototype replaces the individual c_t and combines with each transition's own s_t to form the *subprocess embedding* $\tilde{z}_t = [s_t; \bar{c}_k]$, the input of the two endpoint tasks.

3.2 Training the Embeddings

The loop. Subprocesses are found with c_t, whose meaning derives from tasks defined through them. Each training step first partitions every window with current motion channels, at a per-window tolerance drawn from a window-dependent distribution $p(\epsilon)$ that encourages balanced exposure across subprocess granularities (Appendix A.1.2), then, with the partition fixed, evaluates the three tasks below for every transition and updates the encoder and three task networks. Boundaries receive no gradient; the next training step recomputes them from the updated motion channels. All tasks of a window share its tolerance and prototypes; the expectations below are over the tolerance, the prototype weights and, where present, the corruption of the endpoint.

Next-frame prediction. To ground the motion channel in motion, a network $\mathcal{F}$ with parameters $\Theta_{\mathcal{F}}$ predicts the next grid x_{t+1} from the current grid x_t, the motion prototype $\bar{c}_k$ and ϵ (Ye et al., 2025):

$$\mathcal{L}^{(t)}_{\text{next}} = \mathbb{E}\Big[\mathrm{MSE}\left(\mathcal{F}(x_t, \bar{c}_k, \epsilon; \Theta_{\mathcal{F}}), x_{t+1}\right)\Big], \tag{2}$$

where MSE denotes the mean squared error. Because x_t supplies scene context, the prototype is trained to describe the change to x_{t+1}. Sharing the prototype across a subprocess and resampling its mixture weights encourage a consistent motion description.

Endpoint reconstruction. Endpoint reconstruction trains the subprocess embedding to describe the configuration a subprocess reaches: similar motions can end with the apple in the pot or beside it, and the actions that follow differ. A network $\mathcal{G}$ with parameters $\Theta_{\mathcal{G}}$ reconstructs the clean endpoint grid

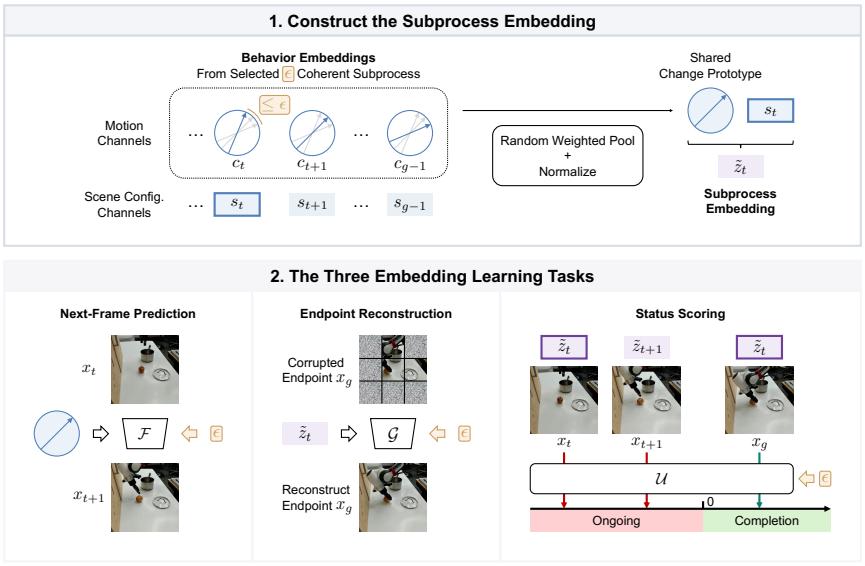


Figure 3: **The three training tasks. (1)** The motion channels of a subprocess are randomly mixed into the prototype $\bar{c}_k$, which joins s_t in the subprocess embedding $\tilde{z}_t = [s_t; \bar{c}_k]$. **(2)** With ϵ, $\mathcal{F}$ predicts x_{t+1} from x_t and $\bar{c}_k$ (left), $\mathcal{G}$ reconstructs the endpoint x_g from its corrupted grid and $\tilde{z}_t$ (middle), and $\mathcal{U}$ scores grids against $\tilde{z}_t$ (right): negative before the endpoint, positive at it, higher for the later of two adjacent ongoing frames. Photographs stand for grids; noisy cells indicate corrupted grid cells.

x_g from a corrupted copy x_g^{noisy}, in which each cell is replaced by Gaussian noise with probability ρ (Appendix A.1.3), together with the subprocess embedding $\tilde{z}_t$ and the tolerance ϵ:

$$\mathcal{L}_{\text{end}}^{(t)} = \mathbb{E}\Big[\text{MSE}\big(\mathcal{G}\big(x_g^{\text{noisy}}, \tilde{z}_t, \epsilon; \Theta_{\mathcal{G}}\big), x_g\big)\Big]. \tag{3}$$

The few surviving cells supply partial scene evidence, as in masked autoencoding (He et al., 2022), from which spatial correlations can be recovered (Bisulco et al., 2025). All transitions of a subprocess share the target x_g. Because the tolerance is resampled and is not an encoder input, the same s_t is supervised using endpoints at multiple granularities, encouraging s_t to encode endpoint information.

Status scoring. Within a subprocess, transitions share the motion prototype and endpoint, but early and late stages can call for different actions. To distinguish these stages, we use temporal position relative to the selected endpoint as a self-supervised cue: current frames are labeled ongoing and the endpoint reached, while an adjacent-order loss encourages later ongoing frames to receive higher scores. This supervision encourages the embedding to capture where the behavior is within the subprocess, beyond its shared motion description. A network $\mathcal{U}$ with parameters $\Theta_{\mathcal{U}}$ scores a frame x_j against the subprocess description $\tilde{z}_t$, $u_{t,j} = \mathcal{U}(x_j, \tilde{z}_t, \epsilon; \Theta_{\mathcal{U}})$ (first index: reference transition; second: scored frame); the score should be negative for ongoing frames and positive at the endpoint (Fig. 3, right). *Completion* asks the current frame to score below $-\delta_{\text{cur}}$ and the endpoint above δ_{end}, with margins $\delta_{\text{cur}}, \delta_{\text{end}} > 0$:

$$\mathcal{L}_{\text{comp}}^{(t)} = \max(0, \delta_{\text{cur}} + u_{t,t}) + \max(0, \delta_{\text{end}} - u_{t,g}) \tag{4}$$

Order asks the later of two adjacent ongoing frames to score higher by a margin $\delta_{\text{order}} > 0$:

$$\mathcal{L}_{\text{order}}^{(t)} = \mathbf{1}\{t+1 < g\} \cdot \max(0, \delta_{\text{order}} - (u_{t+1,t+1} - u_{t,t})) \tag{5}$$

The hinges make pauses and retries cost only in proportion to their violation. The status loss is $\mathcal{L}_{\text{status}}^{(t)} = \mathbb{E}[\mathcal{L}_{\text{comp}}^{(t)} + \mathcal{L}_{\text{order}}^{(t)}]$. Since $\mathcal{U}$ compares a frame with $\tilde{z}_t$, status is relative to that description; Sec. 4.3 tests whether the exported embedding also carries progress on its own.

Regularizer and objective. If all motion channels were identical, every window would collapse into one subprocess at every tolerance. To prevent this, we stack minibatch $\mathcal{B}$'s motion channels as rows of $C_{\mathcal{B}}$ and maximize their coding rate $\widehat{R}(C_{\mathcal{B}})$, a log-determinant measure of feature variation (Yu et al., 2020; Wu et al., 2025, Appendix A.1.4). With weights $\lambda > 0$ and (w, t) indexing the

transitions of the sampled windows, we minimize

$$\mathcal{L}_{\text{emb}}(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{(w,t)\in\mathcal{B}} \left[\lambda_{\text{next}}\mathcal{L}_{\text{next}}^{(w,t)} + \lambda_{\text{end}}\mathcal{L}_{\text{end}}^{(w,t)} + \lambda_{\text{status}}\mathcal{L}_{\text{status}}^{(w,t)}\right] - \lambda_{\text{reg}}\widehat{R}(C_{\mathcal{B}}) \tag{6}$$

over $\Theta_{\mathcal{E}}, \Theta_{\mathcal{F}}, \Theta_{\mathcal{G}}, \Theta_{\mathcal{U}}$, with the image encoder frozen (weights and settings in Appendix A.1.4).

3.3 Policy Learning with Behavior Embedding Prediction

The policy. A policy maps its inputs ξ_t (images and any other context it uses) to features $\pi_h(\xi_t)$, from which it predicts an *action chunk* $\mathbf{a}_{t:t+H-1}$, the next H actions. We call these inputs and the policy's complete training loss $\ell_{\text{base}}^{(t)}$ *native*; we keep both and add one head and one loss.

Targets. We run the frozen Φ_{DC} and $\mathcal{E}$ over each demonstration, camera by camera because $\mathcal{E}$ was trained on single-camera videos, and at each action index t concatenate the V views in a fixed order, $\mathbf{z}_t = [z_t^{(1)}; \ldots; z_t^{(V)}]$ (Appendix A.3.1). Each view contributes the encoder's own output z_t trained under all sampled tolerances. We predict these targets rather than feed them to the policy because they use future frames unavailable at test time.

Head and loss. The head $\mathcal{Q}$, a small multilayer perceptron shared across the H chunk positions, predicts one target per action from the features of the policy's action expert at that position (Liu et al., 2025), encouraging these features to represent behavior throughout the chunk. The mean squared error of these predictions is added to the native loss:

$$\mathcal{L}_{\pi} = \mathbb{E}_{(\eta,t)\sim p_{\pi}}\left[\ell_{\text{base}}^{(t)} + \lambda_{\text{aux}}\,\text{MSE}\left(\mathcal{Q}(\pi_h(\xi_t)), \mathbf{z}_{t:t+H-1}\right)\right], \tag{7}$$

where p_{π} samples a demonstration η and a start t for which the whole chunk has targets, and $\lambda_{\text{aux}} > 0$ weights the auxiliary term. The loss updates $\mathcal{Q}$ and the policy parameters selected for fine-tuning; the targets and their encoders stay fixed. At test time $\mathcal{Q}$, $\mathcal{E}$ and Φ_{DC} are removed, so the policy keeps the inputs, architecture and cost of the base policy. Sec. 4 tests whether the embedding captures the three facts (Sec. 4.3) and whether predicting it helps the policy (Tabs. 1–2).

4 Experiments

Secs. 4.1 and 4.2 test the claim that predicting behavior embeddings makes policies better and more transferable (Sec. 3.3), on vision-language-action (VLA) and world action models (WAM), in simulation under randomization and in the real world under new instructions and visual changes. Sec. 4.3 tests the claim that the embeddings hold the three facts they are trained for (Sec. 3.2): motion, reached configuration, and status, with probes; ablations show what the endpoint and status tasks add.

Behavior Embedding Learning Settings. The encoder is trained on DROID videos (Khazatsky et al., 2024; Appendix A.1) and then frozen. Our Full behavior targets use this one encoder on RoboTwin and Mobile ALOHA data unseen during its training, testing transfer beyond DROID. Probe ablations use separately trained encoders, and comparison targets use their respective encoders. The policies keep their native action and instruction supervision.

4.1 Simulation Policy Evaluation

Settings. We evaluate the VLAs π_0 and $\pi_{0.5}$ and WAM FastWAM (Yuan et al., 2026) on 11 RoboTwin2.0 tasks (Chen et al., 2025c) selected for low reference π_0 clean success (Appendix A.4). Each policy is fine-tuned on 100 demonstrations from a single task (Appendix A.3.3). Variants within each policy share data and optimization; behavior supervision only adds the embedding-prediction loss to the native objective (Sec. 3.3). Targets are computed from the recorded demonstrations, future frames included, and aligned with the chunk positions; the encoders and the head are absent at inference (Appendix A.3.1). We evaluate all policies under clean conditions; clean-trained $\pi_{0.5}$ is also tested without adaptation under randomized background and tabletop textures, distractor clutter, lighting, and table height (Appendix A.4). We report success over 100 rollouts per task and condition and the unweighted task mean. To ask whether the gains come from what our targets encode rather than from any strong visual target, we replace them, under the same clean $\pi_{0.5}$ protocol,

Table 1: **Simulation results.** Success rates (%) on 11 RoboTwin2.0 tasks. Base uses the native objective; other rows add embedding prediction, with **Ours** using our behavior embeddings. Comparison encoders are pretrained; † denotes DROID fine-tuning. The randomized block evaluates clean-trained $\pi_{0.5}$. Bold marks the highest score per policy/evaluation group; underline marks the second-highest distinct score in the clean $\pi_{0.5}$ group, with ties sharing the same mark. Mean weights tasks equally.

Policy	Supervision source	Blocks rank RGB	Blocks rank size	Stack blocks three	Hang mug	Place Object basket	Place Bread basket	Place Dual shoes	Scan object	Place Mouse pad	Place Object scale	Stamp seal	**Mean**
Clean evaluation													
π_0	Base	29.0	15.0	17.0	15.0	53.0	24.0	15.0	20.0	10.0	11.0	20.0	20.8
	Ours	**42.0**	**22.0**	**20.0**	**24.0**	**67.0**	**36.0**	**29.0**	**27.0**	**18.0**	**19.0**	**41.0**	**31.4**
Fast -WAM	Base	71.0	51.0	53.0	44.0	57.0	62.0	66.0	52.0	32.0	26.0	47.0	51.0
	Ours	**82.0**	**62.0**	**80.0**	**64.0**	**72.0**	**75.0**	**89.0**	**66.0**	**37.0**	**44.0**	**48.0**	**65.4**
$\pi_{0.5}$	Base	57.0	25.0	17.0	8.0	47.0	79.0	11.0	23.0	24.0	70.0	48.0	37.2
	Ours	71.0	**55.0**	**30.0**	**15.0**	55.0	**86.0**	**17.0**	**41.0**	**38.0**	**74.0**	70.0	**50.2**
	DINOv2	76.0	40.0	18.0	2.0	65.0	77.0	8.0	27.0	33.0	66.0	56.0	42.5
	DINOv2†	72.0	46.0	20.0	0.0	46.0	84.0	14.0	14.0	32.0	64.0	70.0	42.0
	SigLIP2	63.0	40.0	16.0	5.0	21.0	72.0	16.0	20.0	27.0	67.0	**71.0**	38.0
	SigLIP2†	**82.0**	42.0	23.0	2.0	59.0	80.0	10.0	20.0	28.0	64.0	69.0	43.5
	V-JEPA2.1	80.0	34.0	11.0	3.0	**67.0**	76.0	12.0	37.0	32.0	65.0	62.0	43.5
	V-JEPA2.1†	74.0	28.0	16.0	7.0	24.0	77.0	10.0	36.0	30.0	72.0	70.0	40.4
Randomized evaluation (clean-trained $\pi_{0.5}$)													
$\pi_{0.5}$	Base	24.0	7.0	3.0	3.0	11.0	45.0	**10.0**	4.0	20.0	39.0	**34.0**	18.2
	Ours	**38.0**	**18.0**	**16.0**	**5.0**	**19.0**	**51.0**	**10.0**	**14.0**	**30.0**	**47.0**	**34.0**	**25.6**

by pretrained and DROID-fine-tuned DINOv2 (Oquab et al., 2023), SigLIP2 (Tschannen et al., 2025) and V-JEPA2.1 (Mur-Labadia et al., 2026). DINOv2 and SigLIP2 provide image features; V-JEPA2.1, like ours, uses temporal context (Appendix A.3.4). DROID adaptation precedes frozen extraction and, for SigLIP2, uses the recorded instructions (Appendix A.3.5).

Policy performance and generalization. Behavior supervision improves clean success for all three policies on every task (Tab. 1), supporting the claim in Sec. 3.3 across policy families. FastWAM gains although it already predicts future visual features. Under randomization the clean-trained $\pi_{0.5}$ keeps its advantage, raising mean success and matching or exceeding Base on every task, supporting the hypothesis of transfer to new scenes (Sec. 3).

Visual Representation Comparison. Ours achieves the highest mean success and uniquely improves every task over Base (Tab. 1). The mean-success advantage holds over both image-based targets and temporally contextualized V-JEPA2.1 targets, including their DROID-adapted variants.

4.2 Real-World Policy Evaluation

Settings. On Mobile ALOHA, we compare base $\pi_{0.5}$ with variants supervised by our DC-AE image frontend (+Image), pretrained V-JEPA2.1, DROID-adapted SigLIP2, and our behavior embeddings; the external alternatives were selected for strong simulation performance (Tab. 1). Placement tests contact-rich drawer operation and instruction generalization: the held-out instructions recombine seen objects and destinations. Each variant trains jointly on 56 demonstrations per seen pairing (cup–drawer and cube–container) and is tested on these and held-out cup–container and cube–drawer instructions. Success requires the requested object–destination pairing and drawer closure when applicable. Nesting and folding test visual generalization on precise insertion and deformable manipulation. Nesting requires inserting a doll into a closely fitting small cup, then placing the assembly in a larger cup; 48 demonstrations span two color configurations, and evaluation changes backgrounds or object colors. Cloth folding uses 50 demonstrations across two solid-color cloths to bring one corner toward its diagonal opposite; evaluation introduces new colors and patterns (Fig. 4). Each variant receives 40 trials per condition with a two-minute limit. We report full-task success for placement and nesting and mean final corner-to-target distance for folding (Appendix A.5).

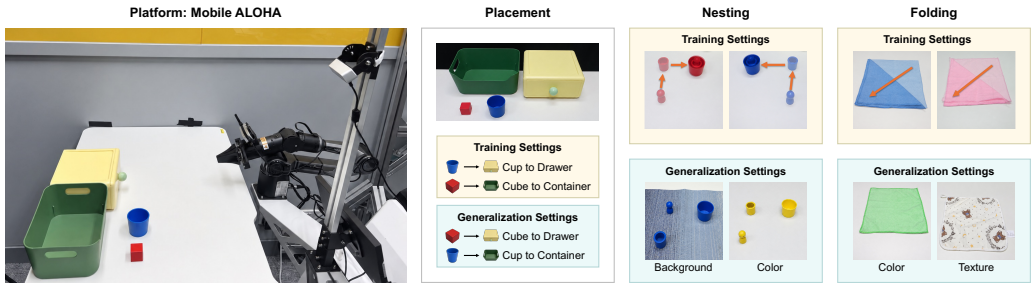


Figure 4: **Mobile ALOHA training and evaluation settings.** Across the three task panels, from left to right: Placement recombines seen objects and destinations; Nesting varies backgrounds and object colors; and Folding evaluates new cloth colors and patterns.

Language generalization. Ours has the highest success on the seen pairings and on the two held-out pairings together (Tab. 2). SigLIP2†, which sees language in pretraining and in DROID adaptation, leads on cup–container, but our variant leads on cube–drawer, where typical failures revert to a seen pairing, e.g., the cube goes into the container after the drawer is opened (Appendix A.5). Embeddings learned without any language thus improve instruction generalization, as predicting where the handled object is going was intended to do (Sec. 3).

Visual generalization. Ours has the highest nesting success and the lowest folding error in every condition, including the standard scene, so the gain is not only robustness: precise insertion and corner alignment improve as well. For +Image and +V-JEPA2.1 on patterned cloth, the 42.4 cm folding error corresponds to the corner-to-target distance when the cloth remains unfolded. Both tasks depend on spatial relations between manipulated objects and their targets, consistent with the information encouraged by endpoint and status supervision (Sec. 3).

Table 2: **Real-world policy results.** Placement and nesting report successful trials / 40 (↑); folding reports mean corner-to-target distance in cm (↓), including no-motion trials. Icons denote cup, cube, container, and drawer. Seen/unseen denotes placement pairings present/absent in fine-tuning. Std. denotes the standard scene; Bg. changes the background; Texture uses patterned cloth; † denotes DROID fine-tuning. Bold marks the best result in each column, including ties.

Supervision	Placement (Seen)		Placement (Unseen)		Nesting			Folding		
	→	→	→	→	Std.	Bg.	Color	Std.	Color	Texture
Base	27/40	29/40	0/40	0/40	14/40	12/40	5/40	2.3	4.3	6.5
+Image	31/40	33/40	14/40	0/40	17/40	12/40	16/40	1.2	3.1	42.4
+V-JEPA2.1	**33/40**	34/40	0/40	0/40	24/40	17/40	17/40	2.3	3.1	42.4
+SigLIP2†	27/40	28/40	**35/40**	5/40	12/40	5/40	4/40	3.0	3.6	4.0
Ours	**33/40**	**37/40**	27/40	**22/40**	**31/40**	**24/40**	**20/40**	**1.1**	**2.5**	**3.5**

4.3 Probing the Information in Behavior Embeddings

Settings. These probes test the second claim: that the exported embedding retains the three kinds of information described in Sec. 3, and reveal what the individual representation-learning objectives contribute. On RoboTwin2.0 ALOHA demonstrations unseen during encoder training, we fit small MLP probes to frozen three-view embeddings from DROID-trained encoders. Full uses all three objectives; w/o End removes endpoint reconstruction and its decoder, while w/o Status removes completion and temporal-order supervision and the status decoder. Further probe and ablation details are given in Appendix A.6.

Skill and execution status. Within-task probes predict normalized episode time as a progress proxy and fine-grained skill labels from recorded manipulation phases (Appendix A.6.3). Full scores highest on time regression for all seven tasks and on skill classification for six (Tab. 3a), although neither label is used in encoder training. Removing either endpoint-related objective usually lowers skill macro-F1, with the largest drops on Hanging Mug for w/o End and Dump Bin Bigbin for w/o Status.

Table 3: **Probing behavior embeddings on RoboTwin2.0. (a)** Within-task normalized-time regression (ten episode folds) and skill classification (macro-F1; five episode folds). **(b)** Grasp-completion transfer: fit on six tasks, evaluate on the seventh. Full uses all three objectives; w/o End and w/o Status omit endpoint reconstruction and status scoring, respectively. Means weight tasks equally. Bold marks the best variant per task and metric. Protocols in Appendix A.6.

(a) Within-task probes on RoboTwin2.0.

Task	Time (R^2 ↑)			Skill (F1↑)		
	w/o End	w/o Status	Full	w/o End	w/o Status	Full
Beat Block Hammer	0.976	0.976	**0.987**	0.824	0.836	**0.852**
Dump Bin Bigbin	0.967	0.970	**0.984**	0.585	0.553	**0.602**
Handover Mic	0.992	0.993	**0.995**	0.925	0.930	**0.944**
Hanging Mug	0.993	0.995	**0.996**	0.642	0.696	**0.723**
Lift Pot	0.926	0.981	**0.985**	0.904	0.936	**0.947**
Put Object Cabinet	0.984	0.980	**0.993**	0.829	0.866	**0.896**
Stack Blocks Three	0.996	0.995	**0.997**	0.718	**0.762**	0.756
Mean	0.976	0.984	**0.991**	0.775	0.797	**0.817**

(b) Grasp-completion transfer.

Held-out task	Macro-F1↑		
	w/o End	w/o Status	Full
Blocks Ranking RGB	0.879	0.867	**0.910**
Blocks Ranking Size	0.863	0.817	**0.886**
Stack Blocks Three	**0.835**	0.821	0.834
Handover Mic	**0.795**	0.609	0.544
Hanging Mug	0.571	0.612	**0.616**
Beat Block Hammer	0.615	0.605	**0.699**
Put Object Cabinet	0.437	0.600	**0.712**
Mean	0.714	0.704	**0.743**

Completion transfer. Using grasp completion as a shared criterion, we fit the probe on six tasks and evaluate it on the seventh (Tab. 3b; Appendix A.6.4). Full leads on five of seven tasks, with its largest margin over both ablations on Put Object Cabinet. The w/o End variant is slightly better on Stack Blocks Three and substantially better on Handover Mic.

Scene configuration and spatial target. On Stack Blocks Three, Full embeddings support decoding grasp/place, block identity, current end-effector position and skill-conditioned spatial targets (Fig. 5). Full reduces validation RMSE for target and current positions by about 70% and 77% relative to either ablation, with comparable block recognition (Appendix A.6.5). These validation results suggest that endpoint reconstruction and status scoring improve the spatial information recoverable by the readout.

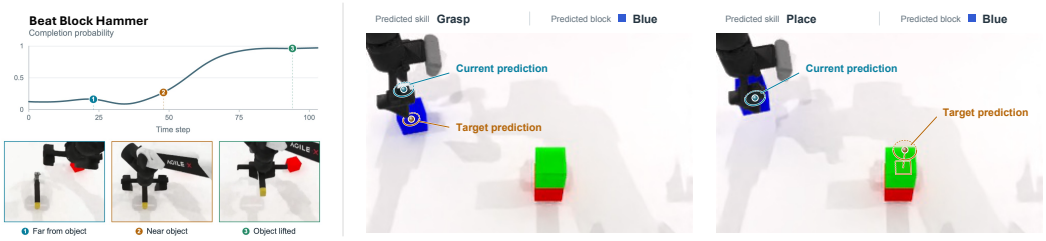


Figure 5: **Left:** grasp-completion probe probability on Beat Block Hammer, with keyframes labeled "Far from object," "Near object," and "Object lifted." **Middle/right:** grasp and place examples on Stack Blocks Three, showing skill, block, current end-effector position and target predictions.

5 Discussion

Behavior embeddings learned from unlabeled videos improve success across VLA and world-action models, with real-world gains under unseen instructions and appearances. Probes recover motion, reached configuration and execution status; ablations support the contributions of endpoint reconstruction and status scoring. Auxiliary prediction adds no inference cost. Three limitations motivate further work. First, encoder training is limited to DROID robot videos; scaling to cheaper, more abundant human data and arbitrary videos remains unexplored. Second, policy integration is limited to auxiliary prediction during fine-tuning; future work could predict intermediate behavior tokens before actions in Transformers, analogous to a chain of thought, or supervise larger models during pretraining. Third, motion-coherence boundaries may miss meaningful subgoals, and tolerance sampling does not ensure balanced exposure to complete partitions. Scene-informed boundaries and balancing across subprocess durations and frequencies merit study. Supplementary success/failure examples motivate encoding collected rollout segments to obtain progress signals for detecting stalled execution and guiding recovery. Future work could use these signals as reward.

AI USE STATEMENT

We used OpenAI Codex to assist with literature search and comparison, manuscript drafting and revision, and improvements to clarity and readability. We also consulted Codex for feedback on experimental methodology and discussion of the interpretation and limitations of results. Additional assistance included checking consistency across mathematical notation, methodological descriptions, and reported results, and preparing LaTeX tables, figures, and supporting scripts. AI-assisted material was revised through author feedback and checked against the manuscript, source materials, and result tables. The authors made the final research decisions and take responsibility for the accuracy, originality, and integrity of the final text, claims, and artifacts.

ETHICS STATEMENT

This work studies representation learning for robotic manipulation using existing robot datasets, simulation, and laboratory evaluations of object placement, nesting, and cloth folding. The intended benefit is more reliable transfer of learned robot behavior across tasks and visual conditions. The evaluated policies can still fail under unfamiliar instructions or environments; the reported performance does not establish safe deployment around people. Deployment would require application-specific risk assessment, human oversight, and appropriate stopping mechanisms. Learned representations may inherit limitations and biases in the objects, environments, and behaviors covered by their source data. Use or redistribution of datasets and pretrained models should respect their licenses and applicable privacy requirements.

REPRODUCIBILITY STATEMENT

Section 3 specifies the representation-learning objectives, subprocess discovery procedure, and integration with policy learning. Appendix A.1 documents video preprocessing, network architectures, sampling, optimization, and training hyperparameters, while Appendix A.3 describes target extraction, temporal alignment, policy adaptation, and comparison encoders. Section 4 and Appendices A.4 and A.5 describe the evaluation tasks, demonstration counts, test conditions, and outcome measures. Appendix A.6 describes the representation probes and associated evaluation protocols. These descriptions identify the datasets and pretrained resources used in the study and distinguish offline representation extraction from policy inference.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Tewodros W Ayalew, Xiao Zhang, Kevin Yuanbo Wu, Tianchong Jiang, Michael Maire, and Matthew R Walter. Progressor: A perceptually guided reward estimator with self-supervised online refinement. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10297–10306. IEEE, 2025.

Anthony Bisulco, Rahul Ramesh, Randall Balestriero, and Pratik Chaudhari. From linearity to non-linearity: How masked autoencoders capture spatial correlations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16441–16450, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 96539–96560, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f062da1973ac9ac61fc6d44dd7fa309f-Paper-Conference.pdf.

Junyu Chen, Dongyun Zou, Wenkun He, Junsong Chen, Enze Xie, Song Han, and Han Cai. Dc-ae 1.5: Accelerating diffusion model convergence with structured latent space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19628–19637, October 2025b.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025c.

Dwip Dalal, Shivansh Patel, Chahit Jain, Jeonghwan Kim, Utkarsh Mishra, Alex Baratian, Hyeonjeong Ha, Heng Ji, Svetlana Lazebnik, and Unnat Jain. Generalizable vla finetuning via representation anchoring and language-action alignment. *arXiv preprint arXiv:2607.13429*, 2026.

Trung Dao, Sankalp Yamsani, Jaden Park, Joohyung Kim, and Yong Jae Lee. Think like a world model, act like a vla: Distilling world-model representations into compact robot policies, 2026. URL https://arxiv.org/abs/2609.24682.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1801–1810. IEEE, 2019.

Hung-Chieh Fang, Kuo-Han Hung, Chu-Rong Chen, Po-Jung Chou, Chun-Kai Yang, Po-Chen Ko, Yu-Chiang Wang, Yueh-Hua Wu, Min-Hung Chen, and Shao-Hua Sun. Learning skills from action-free videos. *arXiv preprint arXiv:2512.20052*, 2025.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 38574–38583, June 2026.

Youhe Feng, Hansen Shi, Haoyang Li, Xinlei Guo, Yang Wang, Chengyang Zhang, Jinkai Zhang, Xiaohan Zhang, Jie Tang, and Jing Zhang. Procvlm: Learning procedure-grounded progress rewards for robotic manipulation. *arXiv preprint arXiv:2605.08774*, 2026.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

Bing Hu, Zaijing Li, Rui Shao, Junda Chen, April Hua Liu, Wei-Shi Zheng, and Liqiang Nie. From abstraction to instantiation: Learning behavioral representation for vision-language-action model. *arXiv preprint arXiv:2605.22671*, 2026.

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.

Guangqi Jiang, Yifei Sun, Tao Huang, Huanyu Li, Yongyuan Liang, and Huazhe Xu. Robots pre-train robots: Manipulation-centric robotic representation from large-scale robot datasets. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yTEwmr1TJb.

Siddharth Karamcheti, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023.

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

Hanjung Kim, Lerrel Pinto, and Seon Joo Kim. Hierarchical latent action model. *arXiv preprint arXiv:2603.05815*, 2026a.

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*, 2026b.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 132324–132345, 2026a.

Jianxiong Li, Jinliang Zheng, Yinan Zheng, Liyuan Mao, Xiao Hu, Sijie Cheng, Haoyi Niu, Jihao Liu, Yu Liu, Jingjing Liu, et al. Decisionnce: Embodied multimodal representations via implicit preference learning. *arXiv preprint arXiv:2402.18137*, 2024.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026b.

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.

Mengya Liu, Baoxiong Jia, Jiangyong Huang, Jingze Zhang, and Siyuan Huang. Lara: Latent action representation alignment for vision-language-action models. *arXiv preprint arXiv:2606.07100*, 2026a.

Ruizhe Liu, Qian Luo, and Yanchao Yang. Infocon: Concept discovery with generative and discriminative informativeness. In *International Conference on Learning Representations*, volume 2024, pp. 37726–37752, 2024.

Ruizhe Liu, Pei Zhou, Qian Luo, Li Sun, Jun Cen, Yibing Song, and Yanchao Yang. Himacon: Discovering hierarchical manipulation concepts from unlabeled multi-modal data. *arXiv preprint arXiv:2510.11321*, 2025.

Yuanzhe Liu, Jingyuan Zhu, Yuchen Mo, Gen Li, Xu Cao, Jin Jin, Yifan Shen, Zhengyuan Li, Tianjiao Yu, Wenzhen Yuan, et al. Palm: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation. *arXiv preprint arXiv:2601.07060*, 2026b.

Kang Luo and Hesheng Wang. Graphpoint: Semantic entity graphs and point trajectories for compositional robot manipulation. *arXiv preprint arXiv:2609.18358*, 2026.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1. 5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pp. 23301–23320. PMLR, 2023.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *Advances in Neural Information Processing Systems*, 37:4062–4089, 2024.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

Vivek Myers, Bill Zheng, Anca Dragan, Kuan Fang, and Sergey Levine. Temporal representation alignment: Successor features enable emergent compositionality in robot instruction following. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=yaS3JWQRQ6.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

NVIDIA, :, Aditi, Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, Aarti Basant, Mukesh Beladiya, Mohammad Qazim Bhat, Zaid Pervaiz Bhat, Dan Blick, Vanni Brighella, Han Cai, Tiffany Cai, Eric Cameracci, Jiaxin Cao, Yulong Cao, Mark Carlson, Carlos Casanova, Ting-Yun Chang, Yan Chang, Yu-Wei Chao, Prithvijit Chattopadhyay, Roshan Chaudhari, Chieh-Yun Chen, Junyu Chen, Ke Chen, Qizhi Chen, Wenkai Chen, Xiaotong Chen, Yu Chen, An-Chieh Cheng, Click Cheng, Xiu Chia, Jeana Choi, Chaeyeon Chung, Wenyan Cong, Yin Cui, Magdalena Dadela, Nalin Dadhich, Wenliang Dai, Joyjit Daw, Alperen Degirmenci, Rodrigo Vieira Del Monte, Robert Denomme, Sameer Dharur, Marco Di Lucca, Ke Ding, Wenhao Ding, Yifan Ding, Yuzhu Dong, Nicole Drumheller, Yilun Du, Aigul Dzhumamuratova, Aleksandr Efitorov, Hamid Eghbalzadeh, Naomi Eigbe, Imad El Hanafi, Hassan Eslami, Benedikt Falk, Jiaojiao Fan, Jim Fan, Amol Fasale, Sergiy Fefilatyev, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Vikram Fugro, Prashant Gaikwad, TJ Galda, Katelyn Gao, Yihuai Gao, Wenhang Ge, Sreyan Ghosh, Arushi Goel, Vivek Goel, Akash Gokul, Rama Govindaraju, Jinwei Gu, Miguel Guerrero, Elfie Guo, Aryaman Gupta, Siddharth Gururani, Hugo Hadfield, Song Han, Ankur Handa, Zekun Hao, Mohammad Harrim, Ali Hassani, Nathan Hayes-Roth, Yufan He, Chris Helvig, Cyrus Hogg, Madison Huang, Michael Huang, Sophia Huang, Yufan Huang, Jacob Huffman, DeLesley Hutchins, Suneel Indupuru, Boris Ivanovic, Arihant Jain, Joel Jang, Ryan Ji, Yanan Jian, Dongfu Jiang, Jingyi Jin, Atharva Joshi, Nikhilesh Joshi, Pranjali Joshi, Andy Ju, Jaehun Jung, Weiwei Kang, Scott Kassekert, Jan Kautz, Ashna Khetan, Julia Kiczka, Slawek Kierat, Gwanghyun Kim, Kuno Kim, Sunny Kim, Kezhi Kong, Xin Kong, Zhifeng Kong, Tomasz Kornuta, Egor Krivov, Hui Kuang, Saurav Kumar, Chia-Wen Kuo, George Kurian, Wojciech Kutak, JF Lafleche, Himangshu Lahkar, Omar Laymoun, Jayjun Lee, Sanggil Lee, Gabriele Leone, Boyi Li, Freya Li, Jiajun Li, Jinfeng Li, Ling Li, Pengcheng Li, Shangru Li, Tingle Li, Xiaolong Li, Xuan Li, Zhaoshuo Li, Zhiqi Li, Hao Liang, Maosheng Liao, Chen-Hsuan Lin, Tsung-Yi Lin, Ming-Yu Liu, Sifei Liu, Zihan Liu, Hai Loc Lu, Xiangyu Lu, Alice Luo, Ruipu Luo, Wenjie Luo, Jiangran Lyu, Martin Ding Ma, Nic Ma, Qianli Ma, Dawid Majchrowski, Louis Marcoux, Miguel Martin, Qing Miao, Ashkan Mirzaei, Shreyas Misra,

Kaichun Mo, Durra Mohsin, Hyejin Moon, Pawel Morkisz, Saeid Motiian, Kirill Motkov, Seungjun Nah, Yashraj Narang, Deepak Narayanan, Thabang Ngazimbi, Julian Ouyang, Shubham Pachori, David Page, Yatian Pang, Sehwi Park, Mahesh Patekar, Mostofa Patwary, Marco Pavone, Trung Pham, Wei Ping, Soha Pouya, Shrimai Prabhumoye, Varun Praveen, Delin Qu, Hesam Rabeti, Morteza Ramezanali, Marilyn Reeb, Xuanchi Ren, Kristen Rumley, Wojciech Rymer, Jun Saito, Yeongho Seol, John Shao, Piyush Shekdar, Tianwei Shen, Humphrey Shi, Min Shi, Stella Shi, Kevin Shih, Mohammad Shoeybi, Mateusz Sieniawski, Shuran Song, Alexander Sotelo, Amir Sotoodeh, Sunil Srinivasa, Vignesh Srinivasakumar, Bartosz Stefaniak, Rahul Heinrich Steiger, Shangkun Sun, Jiaxiang Tang, Shitao Tang, Yangyang Tang, Yue Tang, Tolou Tavakkoli, Kayley Ting, Krzysztof Tomala, Wei-Cheng Tseng, Jibin Varghese, Sergei Vasilev, Thomas Volk, Raju Wagwani, Roger Waleffe, Andrew Z. Wang, Boxiang Wang, Haoxiang Wang, Qiao Wang, Shihao Wang, Shijie Wang, Ting-Chun Wang, Yan Wang, Yu Wang, Rohit Watve, David Wehr, Fangyin Wei, Xinshuo Weng, Jay Zhangjie Wu, Kedi Wu, Hongchi Xia, Summer Xiao, Tianjun Xiao, Kevin Xie, Daguang Xu, Jiashu Xu, Mengyao Xu, Ruqing Xu, Xingqian Xu, Yao Xu, Dinghao Yang, Dong Yang, Hans Yang, Xiaodong Yang, Xuning Yang, Yichu Yang, Yurong You, Zhiding Yu, Hao Yuan, Simon Yuen, Xiaohui Zeng, Pengcuo Zeren, Cindy Zha, Haotian Zhang, Jenny Zhang, Jing Zhang, Liangkai Zhang, Paris Zhang, Shun Zhang, Xuanmeng Zhang, Zhizheng Zhang, Ann Zhao, Yilin Zhao, Yuliya Zhautouskaya, Charles Zhou, Fengzhe Zhou, Shilin Zhu, Yuke Zhu, Dima Zhylko, and Artur Zolkowski. Cosmos 3: Omnimodal world models for physical ai, 2026. URL https://arxiv.org/abs/2606.02800.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Xiaojiang Peng, Kai Peng, Jie Lu, Zheng Lian, Zitong Yu, and Xiaobo Wang. Ac-vla: Robust out-of-distribution action execution via compositional learning. *arXiv preprint arXiv:2607.15714*, 2026.

Jeremy R. Reynolds, Jeffrey M. Zacks, and Todd S. Braver. A computational model of event segmentation from perceptual prediction. *Cognitive Science*, 31(4):613–643, 2007. doi: https://doi.org/10.1080/15326900701399913. URL https://onlinelibrary.wiley.com/doi/abs/10.1080/15326900701399913.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Qin Wang, Abigail Morrison, Hanno Scharr, and Kai Krajsek. Timebridge: Self-supervised video representation learning via start-end joint embedding and in-between frame prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39647–39658, 2026.

Ziyang Wu, Jingyuan Zhang, Druv Pai, Xudong Wang, Chandan Singh, Jianwei Yang, Jianfeng Gao, and Yi Ma. Simplifying DINO via coding rate regularization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68036–68059. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wu25ar.html.

Yifan Xie, Hekun Tian, Jinkun Liu, YuAn Wang, Qiao Sun, and Wenbo Ding. Geolam: Learning geometry-grounded latent actions from unlabeled human videos. *arXiv preprint arXiv:2609.17099*, 2026.

Mengda Xu, Zhenjia Xu, Cheng Chi, Manuela Veloso, and Shuran Song. Xskill: Cross embodiment skill discovery. In *Conference on robot learning*, pp. 3536–3555. PMLR, 2023.

Xiaoxu Xu, Hao Li, Jinhui Ye, Yilun Chen, Jia Zeng, Xinyi Chen, Linning Xu, Dahua Lin, Weixin Li, and Jiangmiao Pang. Futurevla: Joint visuomotor prediction for vision-language-action model. *arXiv preprint arXiv:2603.10712*, 2026.

Haodong Yan, Junfeng Li, Junjie He, Zhide Zhong, MingMing Yu, Wenxuan Song, Jiaguan Zhu, Yangyang Zheng, Yuqiao Du, Jiadi You, et al. Robust-wam: Bridging generative pretraining and semantic foresight in world-action models. *arXiv preprint arXiv:2608.05903*, 2026.

Pengzhi Yang, Xinyu Wang, Pengyu Jing, Kehan Wen, Yiduo Qu, Zhenhao Huang, Minghao Fu, Xin Liu, Yaheng Shen, and Fan Shi. Rarm: Confidence-gated progress reward modeling for rl in manipulation. *arXiv preprint arXiv:2606.22027*, 2026.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations*, volume 2025, pp. 28213–28239, 2025.

Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9422–9434. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6ad4174eba19ecb5fed17411a34ff5e6-Paper.pdf.

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.

Jeffrey M. Zacks, Barbara Tversky, and Gowri Iyer. Perceiving, remembering, and communicating structure in events. *Journal of Experimental Psychology: General*, 130(1):29–58, 2001. doi: 10.1037/0096-3445.130.1.29.

Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pp. 537–546. PMLR, 2022.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, et al. Native video-action pretraining for generalizable robot control. *arXiv preprint arXiv:2607.08639*, 2026.

Han Zhao, Jingbo Wang, Wenxuan Song, Shuai Chen, Yang Liu, Yan Wang, Haoang Li, and Donglin Wang. Frappe: Infusing world modeling into generalist policies via multiple future representation alignment. *arXiv preprint arXiv:2602.17259*, 2026.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

Pei Zhou, Ruizhe Liu, Qian Luo, Fan Wang, Yibing Song, and Yanchao Yang. Autocgp: Closed-loop concept-guided policies from unlabeled demonstrations. In *International Conference on Learning Representations*, volume 2025, pp. 44815–44842, 2025.

A Appendix

The appendix describes behavior embedding learning and its development studies (Secs. A.1 and A.2), policy supervision and comparison encoders (Sec. A.3), simulation and real-world evaluation (Secs. A.4 and A.5), representation probes (Sec. A.6), qualitative subprocess segmentation (Sec. A.7), and additional results (Sec. A.8).

A.1 Behavior Embedding Learning Details

This section specifies the video windows, task networks, sampling procedure and optimization behind Sec. 3. The motion-channel width and endpoint corruption probability are examined in Appendix A.2; the policy auxiliary-loss comparison is in Appendix A.3.2. We use the notation of the main text, omitting the dependence of subprocess indices, prototypes and status scores on the sampled tolerance ϵ.

A.1.1 Video Windows, Encoder Outputs, and Task Networks

Visual input and temporal context. We use all DROID visual demonstration data (Khazatsky et al., 2024), treating each camera stream as a separate single-view video. The frozen DC-AE checkpoint is `mit-han-lab/dc-ae-f32c32-in-1.0-diffusers`. Images are resized from 180×320 to 192×320, encoded as $6 \times 10 \times 32$ latent grids, and cached before embedding training. The cached grids are the DC-AE encoder outputs multiplied by its scaling factor, 0.3189; these scaled grids are used directly as the encoder inputs and prediction targets.[1] No action, proprioception, language, subtask or progress labels enter the embedding-learning objective.

Each source frame is eligible as a window start. A training window contains $T = 60$ frames sampled every five source frames. If fewer sampled frames remain, the last frame retained after sampling is repeated. Padded frames participate in attention, discovery and all three training tasks. The final subprocess ends at the last window frame, whose grid is x_T.

Output indexing and channel roles. The bidirectional trajectory encoder flattens each grid into a 1920-dimensional frame token and uses learned temporal positions. Its first Transformer output position is discarded. The output at position $t + 1$ supplies z_t, aligned with the sampled transition $x_t \to x_{t+1}$, for $t = 1, \ldots, 59$. Every retained output uses the full recorded window and participates in all three training tasks. The coordinate order is

$$z_t = [s_t; c_t], \qquad s_t \in \mathbb{R}^{256}, \quad c_t \in \mathbb{R}^{64}, \qquad \|s_t\|_2 = \|c_t\|_2 = 1.$$

Thus coordinates 1–256 form the scene configuration channel and 257–320 the motion channel.

Conditioning and readout. Each task network processes the grid as 6×10 spatial cells, each represented by a 32-dimensional token, together with tokens encoding the relevant behavior representation and tolerance. The tolerance uses a 64-dimensional sinusoidal encoding. The representation and tolerance also condition feature-wise linear modulation (FiLM), which scales and shifts intermediate attention and feed-forward features. The next-frame predictor $\mathcal{F}$ uses the shared prototype; the endpoint decoder $\mathcal{G}$ and status head $\mathcal{U}$ use $\tilde{z}_t$. The tolerance identifies the grouping criterion used to construct their conditioning representation; the selected endpoint is $g - t$ sampled transitions ahead of t.

A.1.2 Tolerance Sampling, Shared Prototypes, and Gradients

Sampling from window-prefix diameters. For each current 60-frame training window, compute the prefix diameters of its 59 motion channels:

$$d_j = \max_{1 \le p,q \le j} \operatorname{dist}(c_p, c_q), \qquad j = 1, \ldots, 59.$$

A prefix diameter measures all pairwise distances inside that prefix. Let $b_1 < \cdots < b_q$ be the distinct diameters strictly between zero and one, and set $b_0 = 0$, $b_{q+1} = 1$. Discarding repeated cut points and endpoints avoids zero-width intervals. We first select an interval with equal probability, then draw uniformly within it:

$$J \sim \operatorname{Unif}\{0, \ldots, q\}, \qquad \epsilon \mid J = i \sim \operatorname{Unif}[b_i, b_{i+1}). \tag{8}$$

The window-dependent distribution $p(\epsilon)$ in the main text therefore has density

$$p_w(\epsilon) = \sum_{i=0}^{q} \frac{\mathbf{1}\{b_i \le \epsilon < b_{i+1}\}}{(q+1)(b_{i+1} - b_i)}. \tag{9}$$

[1] The factor is specified in MIT HAN Lab's DC-AE release instructions; its latent-generation script multiplies the encoder output by this factor before saving.

Table 4: Behavior embedding learning architectures. The encoder uses a hidden dimension of 2048 and feed-forward layers $2048 \to 8192 \to 2048$. The three task networks use a hidden dimension of 1024 and feed-forward layers $1024 \to 4096 \to 1024$. All Transformers use 16 attention heads, ReLU activations, layer normalization, and no dropout. MLP dimensions are listed in order.

Module	Inputs and output	Architecture and trainable role
DC-AE Φ_{DC}	192×320 RGB image; output grid $6 \times 10 \times 32$.	Frozen pretrained compressor. Hugging Face checkpoint: `mit-han-lab/dc-ae-f32c32-in-1.0-diffusers`.
Trajectory encoder $\mathcal{E}$	$T = 60$ flattened frame grids, each of dimension 1920; output $(T-1) \times 320$, ordered as the 256 scene configuration then the 64 motion dimensions.	Six-layer bidirectional Transformer with learnable temporal positional embeddings. Input MLP: $1920 \to 8192 \to 2048$; output MLP: $2048 \to 8192 \to 320$. The first Transformer output position is discarded; the remaining $T-1$ positions supply the behavior embeddings. Jointly trained with the task networks.
Next-frame predictor $\mathcal{F}$	Current grid x_t, prototype $\bar{c}_k \in \mathbb{R}^{64}$, and tolerance ϵ; output next grid $6 \times 10 \times 32$.	Two-layer Transformer with 71 token positions: 60 spatial, 10 prototype, and one tolerance token. Each spatial output is mapped to 32 latent features by an MLP $1024 \to 4096 \to 32$. Jointly trained shared predictor.
Endpoint decoder $\mathcal{G}$	Corrupted endpoint grid x_g^{noisy}, subprocess embedding $\tilde{z}_t \in \mathbb{R}^{320}$, and tolerance ϵ; output clean grid $6 \times 10 \times 32$.	Two-layer Transformer with 71 token positions: 60 spatial, 10 behavior, and one tolerance token. Each spatial output is mapped to 32 latent features by an MLP $1024 \to 4096 \to 32$. Jointly trained shared decoder.
Status head $\mathcal{U}$	Grid x_j to be scored, subprocess embedding $\tilde{z}_t \in \mathbb{R}^{320}$, and tolerance ϵ; output signed scalar $u_{t,j}$.	Two-layer Transformer with 72 token positions: 60 spatial, 10 behavior, one tolerance token, and one additional score token. The output at the score-token position is mapped to the scalar score by an MLP $1024 \to 4096 \to 1$. Jointly trained shared scorer.

For $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{U}$, an MLP $d_{\mathrm{in}} \to 10240 \to 10240$ maps the respective behavior representation to ten 1024-dimensional tokens, with $d_{\mathrm{in}} = 64$ for $\mathcal{F}$ and $d_{\mathrm{in}} = 320$ for $\mathcal{G}$ and $\mathcal{U}$.

Each interval has probability $1/(q+1)$, regardless of width. With no interior cut points, this becomes uniform sampling on $[0, 1)$. The distribution is recomputed from the current window embeddings.

The sampler balances intervals defined by prefixes beginning at the window start to some extent. A greedy partition also contains subprocesses beginning later; their diameters can introduce additional partition changes inside these intervals. Consequently, equal interval probabilities do not imply equal probabilities for complete partitions. Every interval remains in the sampling support, so partition regimes of positive width can still be sampled. Different tolerances can yield the same partition, and greedy partitions need not be nested.

One prototype per subprocess per step. One tolerance is drawn per window and reused by all three training tasks. Within each discovered subprocess, we draw independent $r_j \sim \mathrm{Unif}(0, 1)$, normalize their weighted sum of motion channels, and reuse the resulting prototype across every member transition and all three objectives. Each member supplies its own current grid for next-frame prediction. For $[a, g)$, the same 64-dimensional prototype supports every mapping $x_j \mapsto x_{j+1}$, $j = a, \ldots, g-1$. New weights are drawn on the next training step, so the predictor sees varying mixtures rather than a single fixed mean. This trains the contributing channels to support prediction throughout the subprocess, without requiring their vectors to be identical.

Tolerance sampling and discrete boundary discovery are not differentiated. The sampled tolerance and boundaries remain fixed during a step. Gradients through the weighted sum and its normalization update its contributing motion channels; endpoint reconstruction and status scoring also update each transition's scene configuration channel. The next encoder pass recomputes the partition from the updated outputs. The encoder receives no tolerance: its original transition outputs learn under the sampled tolerances, with gradients passing through the prototypes. After training, these original outputs are exported as policy targets without a tolerance or prototype. Coding-rate regularization acts on the original motion channels used in discovery.

A.1.3 Endpoint Reconstruction and Status Scoring

Learning the reached configuration. Endpoint reconstruction recovers the selected endpoint from its corrupted grid and $\tilde{z}_t$. At each of the 60 spatial cells i, corruption independently replaces the entire 32-dimensional token:

$$x_{g,i}^{\text{noisy}} = (1 - m_i)x_{g,i} + m_i\eta_i, \qquad m_i \sim \text{Bernoulli}(0.85), \quad \eta_i \sim \mathcal{N}(0, I_{32}). \tag{10}$$

Both the cell-selection mask and Gaussian noise are redrawn for each transition, including transitions sharing an endpoint. Gaussian replacement provides stochastic corruption in the existing dense grid format. This defines a dense denoising task: the decoder recovers corrupted content while preserving the retained scene evidence in a single clean output grid.

Completion and adjacent temporal order. Completion scores the current grid x_t as ongoing and the selected endpoint grid x_g as reached, both conditioned on $\tilde{z}_t$ and the same tolerance. Coherence-derived endpoints are self-supervised training references, motivated by multi-scale event structure and prediction-based segmentation (Zacks et al., 2001; Reynolds et al., 2007). In particular, the final subprocess terminates at the finite window boundary. Completion therefore denotes reaching the selected reference. The margins are $\delta_{\text{cur}} = \delta_{\text{end}} = 1$. The temporal-order margin is $\delta_{\text{order}} = 0.5$. Order compares $u_{t+1,t+1}$ with $u_{t,t}$ only when $t + 1 < g$: the reference transition changes with the scored frame, so each ongoing grid is paired with its own scene configuration channel and the shared prototype. These adjacent comparisons give a number of terms linear in window length. Stationary intervals occur naturally in manipulation videos and are also introduced by repeated-frame padding, which we retain in the objective. During these intervals, temporal reference labels may not be distinguishable from visual state alone. We therefore use completion and temporal order as weak supervisory cues, with soft hinge penalties that allow local violations. The order term is defined as zero when $t + 1 = g$, including at the last transition $t = T - 1$, so no score for an undefined z_T is evaluated.

A.1.4 Loss Reduction, Coding Rate, and Joint Optimization

For N_w windows, the training batch contains $|\mathcal{B}| = 59N_w$ window–transition pairs. All three task losses use a common mean over these pairs, with temporal-order eligibility specified in Eq. 5. Ineligible order entries are zero and remain in the denominator $59N_w$; there is no separate renormalization by the number of eligible comparisons.

Coding-rate regularization is computed separately on each GPU. For the local minibatch $\mathcal{B}$, let $C_{\mathcal{B}} \in \mathbb{R}^{n_c \times d_c}$ stack all its original unit-normalized motion channels, with $n_c = |\mathcal{B}| = 16 \times 59 = 944$. The log-determinant estimator follows coding-rate reduction (Yu et al., 2020) and its use in SimDINO (Wu et al., 2025):

$$\widehat{R}(C_{\mathcal{B}}) = \frac{1}{2} \log\det\left(I_{d_c} + \frac{d_c}{n_c\sigma_{\text{reg}}^2} C_{\mathcal{B}}^\top C_{\mathcal{B}}\right). \tag{11}$$

Here I_{d_c} is the identity, and $\sigma_{\text{reg}} > 0$ controls the distortion scale. We set $\sigma_{\text{reg}}^2 = 0.05$, using 0.05 directly in the denominator. The displayed half-log-determinant is used directly with $\lambda_{\text{reg}} = 0.01$, without an additional exterior scaling factor or cross-GPU gathering of motion channels. The training objective averages these local regularizers across the eight GPUs; the task losses likewise average the local transition losses, giving an overall mean over the 128-window batch. For unit-normalized rows, the second moment has fixed trace; increasing this log determinant favors energy distributed across more directions. The negative sign in the minimized objective rewards this diversity.

One training step encodes the windows, samples a tolerance for each, discovers their subprocesses, and samples one shared prototype per subprocess. It then evaluates all 59 transitions per window, with

independently corrupted endpoints, and updates $\mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{U}$ jointly. The DC-AE stays frozen. The trajectory encoder is initialized randomly. Its linear-layer weights use Xavier uniform initialization, and their biases, where present, are initialized to zero. The checkpoint is saved immediately after the single epoch and then frozen for behavior-target extraction and representation evaluation. The optimization settings are collected in Table 5.

Table 5: Settings for joint behavior embedding learning.

Setting	Value
Task-loss weights $(\lambda_{\text{next}}, \lambda_{\text{end}}, \lambda_{\text{status}})$	$(1, 1, 1)$
Coding-rate weight λ_{reg}	0.01
Status margins $(\delta_{\text{cur}}, \delta_{\text{end}}, \delta_{\text{order}})$	$(1, 1, 0.5)$
Grid-loss reduction	Mean over $6 \times 10 \times 32 = 1920$ entries
Supervised transitions	All 59 transitions per window
Window batch	128 windows; 60 sampled frames per window
Coding-rate batch	Per GPU: 16 windows, 944 motion-channel vectors
Hardware	Eight NVIDIA H100 GPUs; 16 windows per GPU
Training duration / checkpoint	One epoch; approximately two days; checkpoint at the end of the epoch
Coding-rate distortion	$\sigma^2_{\text{reg}} = 0.05$ (SimDINO convention)
Optimizer	AdamW; learning rate 5×10^{-5}; weight decay 0.01
Schedule	Cosine; no warm-up
Tolerance / prototype reuse	One tolerance per window; one prototype per subprocess per step
Endpoint corruption	Probability 0.85; standard Gaussian tokens; redrawn per transition
Latent scaling	DC-AE encoder output multiplied by 0.3189

A.2 Development Studies for Design Choices

We used two simplified DROID training studies to select the motion-channel width and endpoint corruption probability. Each study made one pass over the DROID frames used for that study. Their probe results describe these development models; the final representation uses the joint objective in Sec. 3.2.

Shared probe protocol. The development probes use 100 episodes from the RoboTwin Blocks Ranking RGB training demonstrations, separate from policy evaluation rollouts. We split these episodes into ten fixed episode-level folds with seed 42, fitting on 90 complete episodes and evaluating on the remaining 10 in each fold. All candidates within a study use the same episode assignments. The representation encoders are trained on DROID and frozen for probing. Features from the synchronized head, left and right camera streams are extracted independently and concatenated in the order of Table 8. Each feature coordinate is standardized using the mean and standard deviation fitted on the training episodes of that fold; the same statistics are applied to the held-out episodes. Regression targets are standardized separately with training-fold statistics.

The probes follow the common regression fitting settings in Appendix A.6.1, while retaining their pilot-specific widths. We fit a separate scalar regressor for each target coordinate. Each probe has two linear layers with ReLU between them: $d_{\text{in}} \to 256 \to 1$, where d_{in} is the concatenated input width. Fitting uses squared-error regression with Adam, learning rate 10^{-3}, weight decay 10^{-4}, batch size 256, and at most 500 iterations with early stopping. Each coordinate's R^2, and Pearson r where reported, is computed on each held-out fold and then averaged over the ten folds.

Motion-channel width. The width study used next-frame prediction alone, with a motion channel and no scene configuration channel, and assessed the representation by regressing action position changes. Concatenating the three camera views gives $d_{\text{in}} = 3d_c$, or 96, 192, 384 and 768 coordinates for the four tested widths. Table 6 reports the six translation coordinates of the two arms. Width 64 attains the highest R^2 and Pearson r on five of six coordinates; width 128 leads on left-arm Δx. These results support the adopted $d_c = 64$ for the motion channel.

Table 6: Motion-channel width pilot on Blocks Ranking RGB. Each metric is shown separately; bold indicates the best value in each coordinate. Only next-frame prediction was used for representation training.

d_c	Metric	Left Δx	Left Δy	Left Δz	Right Δx	Right Δy	Right Δz
32	R^2	0.9066	0.9350	0.7727	0.9071	0.9318	0.8245
	Pearson r	0.9522	0.9670	0.8791	0.9525	0.9654	0.9082
64	R^2	0.9210	**0.9518**	**0.8477**	**0.9365**	**0.9560**	**0.8899**
	Pearson r	0.9605	**0.9756**	**0.9207**	**0.9679**	**0.9781**	**0.9436**
128	R^2	**0.9233**	0.9350	0.8256	0.9220	0.9451	0.8663
	Pearson r	**0.9611**	0.9670	0.9086	0.9606	0.9722	0.9312
256	R^2	0.9006	0.9161	0.7467	0.9034	0.9315	0.7981
	Pearson r	0.9491	0.9574	0.8645	0.9512	0.9652	0.8935

Endpoint corruption probability. A separate pilot mapped a single image's DC-AE grid to a 320-dimensional representation with a Transformer encoder. A decoder reconstructed the clean DC-AE grid from that representation and a corrupted copy of the grid. Concatenating the three camera representations gives $d_{\text{in}} = 960$ for probing. We evaluated scene information by regressing the positions (x, y, z) and four quaternion components (r_1, r_2, r_3, r_4) of each of three blocks. Table 7 reports R^2 for all 21 coordinates.

For each coordinate, the six probabilities are ranked from highest to lowest R^2, receiving scores $5, 4, 3, 2, 1, 0$. Writing $\text{rank}_j(\rho) \in \{1, \dots, 6\}$ for the descending rank in coordinate j, the aggregate is

$$S(\rho) = \sum_{j=1}^{21} \big(6 - \text{rank}_j(\rho)\big).$$

All 21 columns have distinct values at the reported precision. The totals are 34, 47, 71, 51, 67 and 45 for $\rho = 0.75, 0.80, 0.85, 0.90, 0.95, 1.00$, respectively. Thus $\rho = 0.85$ receives the highest score, 71, followed by 0.95 with 67. The selection uses ranks across coordinates rather than an average of their raw R^2 values.

A.3 Policy Targets and Auxiliary Prediction

A.3.1 Target Generation and Action Alignment

A forward window for every action index. For each task, let $L_{\min}$ be the shortest demonstration length in frames in that task's policy-training set. We compute this minimum separately for each simulation and real-world task. The labeling stride is $f_{\text{lab}} = \max\{1, \lfloor L_{\min}/60 \rfloor\}$, so it stays positive even when the shortest demonstration has fewer than 60 frames. For a demonstration with T_π frames, let

$$\iota_t(j) = \min\{t + j f_{\text{lab}}, T_\pi\}, \qquad j = 0, \dots, 59.$$

For every action start $t = 1, \dots, T_\pi - 1$ and view v, the frozen encoders produce

$$\begin{aligned} W_t^{(v)} &= (x^{(v)}_{\iota_t(0)}, \dots, x^{(v)}_{\iota_t(59)}), \\ (\zeta^{(v)}_{t,1}, \dots, \zeta^{(v)}_{t,59}) &= \mathcal{E}(W_t^{(v)}), \qquad z_t^{(v)} = \zeta^{(v)}_{t,1}. \end{aligned} \tag{12}$$

Indices beyond the episode end repeat the final source frame. The first retained embedding labels action $\mathbf{a}_t$; each successive action index gets its own forward window. Outputs of overlapping windows are not averaged. The final frame has no action and requires no policy target.

For example, $L_{\min} = 600$ gives $f_{\text{lab}} = 10$. Action 20 uses the first embedding from frames $20, 30, 40, \dots$, while action 21 uses a new window beginning $21, 31, 41, \dots$. The second output of the first window begins at frame 30 and is not the label for action 21. The embedding-training stride of five and the collection-dependent labeling stride are separate settings. A labeling window spans $59 f_{\text{lab}}$ frame intervals before padding.

Table 7: Corruption-probability pilot: R^2 for three-block pose regression. $r_1, \ldots, r_4$ denote quaternion components. Bold and underline indicate the best and second-best value in each column. The final panel sums the per-coordinate rank scores, from 5 for the best to 0 for the worst.

Block 1

ρ	x	y	z	r_1	r_2	r_3	r_4
0.75	0.9889	0.9788	0.9819	0.9656	0.5294	**0.7689**	0.8620
0.80	0.9911	**0.9795**	<u>0.9826</u>	**0.9667**	0.5448	<u>0.7429</u>	0.8816
0.85	**0.9966**	0.9721	**0.9831**	0.9596	0.5377	0.7164	**0.9041**
0.90	<u>0.9913</u>	<u>0.9793</u>	0.9803	<u>0.9658</u>	0.5814	0.7245	<u>0.8931</u>
0.95	0.9904	0.9730	0.9822	0.9556	<u>0.6079</u>	0.7398	0.8848
1.00	0.9894	0.9781	0.9801	0.9587	**0.6132**	0.7231	0.8817

Block 2

ρ	x	y	z	r_1	r_2	r_3	r_4
0.75	0.9821	0.9704	0.9753	0.8587	0.5541	0.4946	0.6753
0.80	0.9827	0.9701	0.9779	0.8437	0.5720	0.5547	0.7465
0.85	**0.9862**	0.9719	<u>0.9794</u>	0.8685	**0.6336**	**0.6101**	0.7368
0.90	0.9804	0.9696	0.9792	0.8535	<u>0.6052</u>	0.5840	<u>0.7590</u>
0.95	0.9780	**0.9735**	**0.9802**	**0.8737**	0.5805	0.5833	0.7395
1.00	<u>0.9850</u>	<u>0.9734</u>	0.9784	<u>0.8729</u>	0.5098	<u>0.5949</u>	**0.7596**

Block 3

ρ	x	y	z	r_1	r_2	r_3	r_4
0.75	<u>0.9866</u>	0.9396	0.9819	0.7358	0.4694	0.5589	<u>0.7531</u>
0.80	0.9800	0.9348	0.9805	0.7513	0.4795	**0.5903**	0.7393
0.85	0.9853	**0.9479**	0.9832	<u>0.7703</u>	<u>0.4870</u>	0.5687	**0.7553**
0.90	0.9849	0.9442	0.9823	0.7491	0.4657	<u>0.5877</u>	0.7216
0.95	**0.9871**	<u>0.9463</u>	**0.9846**	**0.7845**	**0.5213**	0.5661	0.7501
1.00	0.9850	0.9389	<u>0.9843</u>	0.7474	0.4556	0.5498	0.7452

Aggregate selection score

ρ	0.75	0.80	0.85	0.90	0.95	1.00
$S(\rho)$	34	47	**71**	51	67	45

Camera concatenation and action chunks. The encoder was trained on single-camera videos, so each camera is encoded independently before concatenation. The per-view target is $z_t^{(v)} = [s_t^{(v)}; c_t^{(v)}]$, and $\mathbf{z}_t$ concatenates these vectors in the order shown in Table 8. Write d_{tar} for the full width of the selected multiview target; for behavior embeddings, $d_{\text{tar}} = Vd_z$. An action chunk $\mathbf{a}_{t:t+H-1}$ is paired with $\mathbf{z}_{t:t+H-1}$; each position is supervised at the same native action index. The indexing is independent of the stride inside each target-generation window. In particular, FastWAM's chunk of 32 action positions receives targets $\mathbf{z}_t, \ldots, \mathbf{z}_{t+31}$; the stride used to construct each video window does not change these action indices.

Table 8: Camera order for behavior targets. Each view is encoded independently, then concatenated in the listed order. The base policy retains its configured inputs.

Platform	Ordered camera keys	d_{tar}
RoboTwin	`head_camera`, `left_camera`, `right_camera`	960
Mobile ALOHA	`observation.images.cam_high`, `observation.images.cam_right_wrist`	640

A.3.2 Auxiliary Prediction Head and Loss

Prediction head. The head reads the fourth-from-last action-expert Transformer hidden layer at each action position; its input is normalized with RMSNorm. For a hidden feature vector h_{ae} of width D_{ae}, the head applies input RMSNorm followed by two biased linear layers with ReLU between them:

$$h_{\text{ae}} \longmapsto \text{Linear}_{m \to d_{\text{tar}}}(\text{ReLU}(\text{Linear}_{D_{\text{ae}} \to m}(\text{RMSNorm}(h_{\text{ae}})))), \quad m = \left\lfloor \frac{D_{\text{ae}} + d_{\text{tar}}}{2} \right\rfloor.$$

There is no normalization between the linear layers, and the output is unnormalized. The same head weights are shared across chunk positions. The released π_0, $\pi_{0.5}$ and FastWAM action experts have feature width $D_{\text{ae}} = 1024$.[2] For behavior targets, the auxiliary MLP dimensions are $1024 \to 992 \to 960$ for RoboTwin and $1024 \to 832 \to 640$ for Mobile ALOHA.

Auxiliary loss and native training inputs. MSE is the auxiliary loss for all targets in the main policy comparisons, including FastWAM and the comparison encoders. For predicted vectors $\widehat{z}_{t,h}$ and frozen behavior targets, with chunk position h counted from zero,

$$\text{MSE}(\widehat{Z}_t, \mathbf{z}_{t:t+H-1}) = \frac{1}{H d_{\text{tar}}} \sum_{h=0}^{H-1} \|\widehat{z}_{t,h} - \mathbf{z}_{t+h}\|_2^2, \tag{13}$$

also averaged across examples in the policy batch. Comparison targets use the same reduction with their respective d_{tar}. The elementwise mean weights coordinates uniformly. The native objective and its input construction remain in place. For flow-based action learning (Black et al., 2024; Intelligence et al., 2025), the action-expert features are evaluated with the native noised action sequence and flow-time conditioning as well as observation and task context. FastWAM retains its native dense visual prediction objective alongside action learning. The added loss trains the auxiliary head and the policy modules selected for adaptation; the stored targets and their encoders remain fixed.

Choice of auxiliary loss. We compared MSE with channelwise cosine loss for FastWAM on RoboTwin Blocks Ranking Size (`blocks_ranking_size`). The comparison uses the same training and evaluation settings, including 100 training demonstrations and 100 clean evaluation rollouts per variant. The cosine alternative averages six losses: one for each separately normalized scene configuration and motion channel from each of the three camera views. Writing $\widehat{z}_{t,h}^{(v)} = [\widehat{s}_{t,h}^{(v)}; \widehat{c}_{t,h}^{(v)}]$, its loss at an aligned action position is

$$\ell_{\cos}^{(t,h)} = \frac{1}{6} \sum_{v=1}^{3} \left[1 - \cos(\widehat{s}_{t,h}^{(v)}, s_{t+h}^{(v)}) + 1 - \cos(\widehat{c}_{t,h}^{(v)}, c_{t+h}^{(v)}) \right]. \tag{14}$$

The loss is averaged over chunk positions and batch examples, with the same way of adaptive loss (Sec. A.3.3). MSE achieves 62.0% success, compared with 51.0% for cosine, an observed improvement of 11.0 percentage points (Table 9). At the tested auxiliary weight, MSE gives higher success in this comparison and is used for the reported policies. Equal loss coefficients do not equalize the numerical scales of coordinate-averaged MSE and channel-averaged cosine loss; this experiment compares the two specified training configurations.

Table 9: Auxiliary-loss comparison for FastWAM on Blocks Ranking Size. All other training and evaluation settings are shared. Each success rate uses 100 clean evaluation rollouts.

Auxiliary loss	Success rate (%)
Channelwise cosine	51.0
MSE (default)	**62.0**

[2] The OpenPI configuration selects the `gemma_300m` action expert; its standard and LoRA variants have width 1024. The FastWAM action-model configuration also specifies width 1024.

A.3.3 POLICY FINE-TUNING SETTINGS

All simulation variants use single-task fine-tuning for 30,000 optimizer updates with a batch size of 32 action chunks. We use LoRA for π_0 and $\pi_{0.5}$; FastWAM adaptation follows its released configuration with these training overrides. Real-world $\pi_{0.5}$ uses full-parameter fine-tuning. Tables 10 and 11 give the chunk lengths, auxiliary weights and study configurations.

Real-world optimization. We initialize from OpenPI's $\pi_{0.5}$ base checkpoint and follow its ALOHA full-fine-tuning defaults, training for 30,000 optimizer updates. The real-world batch size is 32 action chunks. AdamW uses $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon_{\text{Adam}} = 10^{-8}$, weight decay 10^{-10}, and global gradient-norm clipping at 1.0. The learning rate warms up for 1,000 updates to 2.5×10^{-5}, then follows cosine decay to 2.5×10^{-6} at the 30,000-update schedule horizon. The configuration uses an exponential-moving-average decay of 0.99. [3] The base policy and auxiliary-supervision variants share these settings.

Chunk execution. At evaluation time, every policy executes its complete predicted action chunk before the next inference call, in both simulation and the real world. The number of executed actions is therefore the corresponding H in Table 10.

Adaptive auxiliary weight. Instead of a fixed λ_{aux}, we set it adaptively at each training step so that the weighted auxiliary loss stays comparable to the action loss: $\lambda_{\text{aux}}\mathcal{L}_{\text{aux}} \approx \rho\,\mathcal{L}_{\text{action}}$, i.e. $\lambda_{\text{aux}} = \rho \cdot \mathcal{L}_{\text{action}}/\mathcal{L}_{\text{aux}}$, with ρ=0.2. The adaptive weight is treated as a constant during backpropagation. This removes manual retuning of λ_{aux} across different auxiliary targets (e.g., behavior, V-JEPA2.1, SigLIP2, DINOv2).

Table 10: Auxiliary settings. Batch size is 32 action chunks, and all main policy comparisons use MSE. We use an adaptive auxiliary loss weight λ_{aux} with ρ=0.2.

Policy	Setting	Chunk length H
π_0	Simulation	50
$\pi_{0.5}$	Simulation	16
FastWAM	Simulation	32
$\pi_{0.5}$	Mobile ALOHA, behavior / V-JEPA2.1 / SigLIP2 targets	50
$\pi_{0.5}$	Mobile ALOHA, DC-AE +Image	50

Table 11: Policy-study configurations. Demonstration counts describe fine-tuning data; rollout counts describe evaluation.

Study	Policy/adaptation	Fine-tuning data	Evaluation
Simulation policy comparison	π_0, $\pi_{0.5}$: LoRA; FastWAM: released adaptation; single-task	100 demonstrations per task	100 rollouts per variant, task and condition
Simulation target comparison	$\pi_{0.5}$ LoRA	100 clean demonstrations per task; three views	100 clean rollouts per target and task
Real-world policy comparison	$\pi_{0.5}$; full-parameter fine-tuning	112 placement, 48 nesting, 50 folding demonstrations	40 trials per variant and condition; two-minute limit

A.3.4 REPRESENTATION TARGETS AND UPSTREAM RESOURCES

Selection at the policy interface. We prioritize the size of the output representation supplied to the policy when selecting comparison checkpoints. Each action position predicts one vector per camera view, with the views concatenated into a single target. This output budget determines the width of the prediction problem, the size of stored targets, and the dimension-dependent auxiliary

[3] The adopted defaults are documented in the OpenPI ALOHA training configurations and optimizer and learning-rate definitions.

head. Accordingly, we select small/base releases with modest native feature widths, using the extraction procedures below. The target-generation encoders operate offline before policy fine-tuning. Table 12 reports the selected outputs: our 320-dimensional per-view target is smaller than the 384- and 768-dimensional reference targets.

Table 12: Encoders and exported policy targets. Widths are unchanged by DROID adaptation. RoboTwin concatenates three camera views; Mobile ALOHA concatenates the overhead and right-wrist views. A dash denotes a target not evaluated on that platform. Mobile ALOHA uses DROID-adapted SigLIP2 and pretrained V-JEPA2.1.

Representation	Input and exported feature	Per view	RoboTwin	Mobile ALOHA
Behavior embeddings	Video window; full $z_t = [s_t; c_t]$	320	960	640
DINOv2-S/14	Single image; CLS token	384	1152	–
SigLIP2-B/16	Single image; attention-pooled vector	768	2304	1536
V-JEPA2.1-B/16	64-frame video; spatial pooling, first temporal position	768	2304	1536
+Image (DC-AE)	Single image; flattened latent grid	1920	–	3840

All Mobile ALOHA targets concatenate the overhead and right-wrist views, in that order, using the camera keys in Table 8. Behavior targets contain $2 \times 320 = 640$ coordinates. V-JEPA2.1 and SigLIP2 each provide $2 \times 768 = 1536$ coordinates and use an auxiliary MLP of width $1024 \to 1280 \to 1536$, following the common head rule in Appendix A.3.2. Both use MSE with adaptive λ, as specified in Table 10.

Direct image supervision in the real world. The +Image condition uses the frozen DC-AE frontend directly. For each action index, the current frame from each of the overhead and right-wrist views is resized and encoded as in Sec. A.1.1, including the 0.3189 latent scaling factor. Each $6 \times 10 \times 32$ grid is flattened into 1,920 coordinates; concatenating the two views in the order of Table 8 gives a 3,840-dimensional target. Targets are aligned to the same native action positions as the behavior targets, using the current frame at each position. The auxiliary head uses the RMSNorm–MLP construction in Sec. A.3.2, with dimensions $1024 \to 2432 \to 3840$, and predicts all $H = 50$ targets in the chunk. The loss is elementwise MSE averaged over target coordinates, chunk positions and batch examples, weighted by adaptive λ. This condition evaluates direct supervision by the spatial image features supplied to the trajectory encoder.

Image and video features. The DINOv2 checkpoint is `facebook/dinov2-small`. For SigLIP2, we use `google/siglip2-base-patch16-224`. Both encode an individual image at each sampled timestamp, providing references learned through image self-supervision and image–language pretraining, respectively. Their DROID adaptations also operate on individual frames, as detailed below. V-JEPA2.1 is pretrained on images and videos and is used here as a video encoder: we encode 64 sampled frames, pool spatial tokens at each encoded temporal position, and select the first temporal position to obtain one 768-dimensional feature per camera. That feature is computed with temporal context from the clip. Our encoder likewise uses a temporal window, learning transition-aligned targets through next-frame prediction, endpoint reconstruction, and status scoring. This set therefore includes both image-based supervision and a temporally contextualized reference for the same policy prediction task.

V-JEPA2.1 checkpoint and temporal alignment. We use the distilled ViT-B/16 checkpoint `vjepa2_1_vitb_dist_vitG_384.pt`.[4] The DROID-adapted variant starts from this same checkpoint. Both pretrained and DROID-adapted V-JEPA2.1 use the same target-extraction procedure. Labeling follows the forward-window construction in Sec. A.3.1, with 64 frames in place of 60. For the shortest demonstration length $L_{\min}$ in each task's training set, the stride is $f_{\mathrm{VJ}} = \max\{1, \lfloor L_{\min}/64 \rfloor\}$. At action index t, the clip contains frames at $\min\{t + jf_{\mathrm{VJ}}, T_\pi\}$, $j = 0, \ldots, 63$, with the last source frame repeated beyond the episode end. Spatial pooling at the first encoded temporal position supplies the target for that action. A new forward clip is formed at every

[4]The released checkpoint is available from the official V-JEPA2.1 model download.

native action index; later temporal outputs are not used as labels for the successive action indices. Thus both video encoders supply offline targets aligned to the action chunk, despite their different clip lengths and representations. The main policy comparisons use these pooled features without PCA. The separate width-matched PCA variant is described in Appendix A.8.

A.3.5 DROID Adaptation of Comparison Encoders

DROID-FT († in the main tables) denotes adaptation of the representation encoder before frozen policy-target extraction. We initialize DINOv2-S from `facebook/dinov2-small` and SigLIP2-B from `google/siglip2-base-patch16-224`. Both image-encoder recipes sample individual frames from the same configured DROID WebDataset collection. DINOv2 uses images alone; SigLIP2 also uses each frame's recorded `language_instruction`. Table 13 summarizes the adaptation settings.

Table 13: DROID adaptation configurations for the comparison encoders. Batch sizes count source images for DINOv2 and SigLIP2, and video clips for V-JEPA2.1, per optimizer update. DINOv2 produces two augmented views of each image. Feature dimensions are per camera before camera-view concatenation for policy supervision.

Setting	DINOv2-S	SigLIP2-B	V-JEPA2.1
Policy-target feature	CLS token, 384 dimensions	Attention-pooled image vector, 768 dimensions	Spatially pooled first temporal output, 768 dimensions
Adaptation supervision	Cross-view CLS self-distillation	Multi-positive image–instruction sigmoid loss	Masked latent prediction
Trainable encoder modules	Final two ViT blocks and final LayerNorm	Final two vision blocks, post-LayerNorm and attention-pooling head	Full ViT-B/384 context encoder
Other trainable modules	CLS projection head	Logit scale and bias	Predictor
Frozen modules	Earlier backbone blocks and input embeddings; teacher receives EMA updates	Earlier vision blocks, input embeddings and complete text tower	EMA target encoder
Input resolution	224×224	224×224	384×384, 64 frames
Batch / accumulation	256 / 1	1,024 / 1	2 / 1
Optimizer	AdamW, $\beta = (0.9, 0.999)$	AdamW, $\beta = (0.9, 0.95)$	AdamW, $\beta = (0.9, 0.999)$
Learning rate	10^{-5} backbone; 10^{-4} projection head	2.5×10^{-6}	$10^{-4} \rightarrow 6 \times 10^{-4}$
Weight decay / gradient clip	0.04 / 3.0	0.05 / 1.0	0.04 / none
Configured updates / warmup	50,000 / 2,000	10,000 / 500	94,500 / 12,000
LR decay / minimum ratio	Cosine / 0.05	Cosine / 0.05	Cosine / 1.0
Random crop area range	$[0.5, 1.0]$	$[0.7, 1.0]$	$[0.3, 1.0]$
Horizontal flips / seed	Disabled / 42	Disabled / 42	Enabled / 239

DINOv2 adaptation. For each frame, two independently augmented views are encoded by a student and an exponential-moving-average teacher. A projection head maps the 384-dimensional CLS feature through widths 1,024 and 256 to 4,096 logits, with L2 normalization before the final projection. The student distribution for each view matches the centered teacher distribution for the other view using cross-entropy, averaged across the two directions and source images. The student temperature is 0.1; the teacher temperature increases linearly from 0.04 to 0.07 over 5,000 updates. Teacher momentum increases linearly from 0.996 to 1.0, and the logit-center momentum is 0.9. The projection head supplies adaptation supervision; the policy target is the student backbone's CLS feature.

SigLIP2 adaptation. Instructions are stripped of redundant whitespace and deduplicated within each batch. Each unique instruction forms a positive pair with every image carrying that instruction and a negative pair with the other images. The loss sums the signed sigmoid negative log-likelihood over images and averages over unique instructions. The text tower remains frozen, while the selected vision modules and logit scale and bias are updated. Tokenization uses at most 64 tokens. Samples without a nonempty instruction are omitted, and training skips batches containing fewer than two distinct instructions. This recipe supplies image–instruction adaptation; it does not introduce subprocess endpoints or status labels.

V-JEPA2.1 adaptation. V-JEPA2.1 is adapted on DROID video clips using its masked latent-prediction objective. The context encoder and predictor learn to predict layer-normalized latent targets produced by an exponential-moving-average target encoder. The full ViT-B/384 context encoder and predictor are optimized, while the target encoder remains gradient-frozen and is updated only through EMA. The action-based auxiliary loss is disabled. Training uses 64-frame clips at 384×384, AdamW with batch size 2 and weight decay 0.04, and a learning rate warmed from 10^{-4} to 6×10^{-4}. Random resized cropping and horizontal flipping are applied during training. We train for 94,500 optimizer updates and use the final checkpoint for frozen policy-target extraction.

Frame sampling and augmentation. The DINOv2 and SigLIP2 loaders resample shards and shuffle frames, using 16 workers and shuffle buffers of 10,000 for DINOv2 and 20,000 for SigLIP2. DINOv2 applies color jitter, grayscale and asymmetric blur/solarization to its two crops; SigLIP2 uses mild color jitter and a crop aspect ratio in $[0.9, 1.1]$. Image normalization uses the selected pretrained processor. Both image-encoder configurations select automatic mixed precision and disable gradient checkpointing. The training loops update the cosine learning-rate factor after each optimizer step; the first update uses the initialized learning rate.

The recipes adapt both image models to robot demonstrations while retaining their respective supervision: visual consistency for DINOv2 and image–instruction alignment for SigLIP2. They use different data filtering, batch sizes, and update budgets.

Frozen policy-target extraction. All adapted encoders are frozen before policy-target extraction, and each camera view is processed independently before concatenation in the policy target's camera order. DINOv2 and SigLIP2 convert inputs to RGB and use the pixel mean and standard deviation of their corresponding pretrained processors. DINOv2 resizes the shorter image edge to 224 with bicubic interpolation, takes a 224×224 center crop, and returns the student backbone's final 384-dimensional CLS feature without additional normalization. SigLIP2 uses an antialiased bicubic resize to 224×224 without a subsequent crop and returns an L2-normalized, 768-dimensional attention-pooled image feature. For V-JEPA2.1, RGB frames are resized to 384×384 with bilinear interpolation, scaled to $[0, 1]$, and normalized using the ImageNet mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$.

V-JEPA2.1 uses the 64-frame forward windows, task-dependent stride, and first-temporal-output spatial pooling specified in Appendix A.3.4. Each view contributes 768 coordinates for both pretrained and DROID-adapted encoders. Three views give a 2,304-dimensional RoboTwin target; the overhead and right-wrist views give a 1,536-dimensional Mobile ALOHA target.

All extracted features are cached in float32. CUDA extraction uses automatic mixed precision; DINOv2 and SigLIP2 use bfloat16 when supported and float16 otherwise, while V-JEPA2.1 uses bfloat16. The resulting per-camera feature widths are 384 for DINOv2 and 768 for both SigLIP2 and V-JEPA2.1.

A.4 Simulation Policy Experiments

Task selection and fine-tuning. We evaluate 11 selected RoboTwin2.0 tasks with 100 demonstrations per task and single-task policy fine-tuning. The selected tasks have RGB single-task π_0 clean success below 20% in the public leaderboard reference, defining a challenging subset for policy learning. The tasks are Blocks Ranking RGB, Blocks Ranking Size, Stack Blocks Three, Hanging Mug, Place Object Basket, Place Bread Basket, Place Dual Shoes, Scan Object, Place Mouse Pad, Place Object Scale, and Stamp Seal, in the column order of Table 1. The reference leaderboard uses 50 demonstrations per task; our policy results use 100. Its scores guide task selection and are distinct

from the base-policy scores in our comparisons.[5] Each policy is adapted to one task at a time, while transfer inherited from policy pretraining and the DROID-trained targets remains part of the setting.

Evaluation conditions and aggregation. All three policies are evaluated under clean conditions. The clean-trained $\pi_{0.5}$ checkpoints are additionally evaluated without further adaptation using RoboTwin2.0's `demo_randomized.yml` domain-randomization settings. These change background and tabletop textures, add distractor clutter, vary lighting, and perturb table height. The configuration sets `random_table_height` to 0.03 m, `clean_background_rate` and `crazy_random_light_rate` to 0.02, and `random_head_camera_dis` to zero.[6] The separate auxiliary-target comparison uses $\pi_{0.5}$ LoRA, `demo_clean_100`, and three camera views on the same 11 tasks. Each variant uses 100 evaluation rollouts per task and condition. We report task success rates and the unweighted mean over tasks; percentage-point gains are computed before rounding the means. The 100-rollout convention agrees with the public benchmark setting. For task j, the success percentage is $S_j = n_j \times 100\%/100$, where n_j is the number of successful rollouts. The reported mean is $\bar{S} = \frac{1}{11}\sum_{j=1}^{11} S_j$. With equal rollout counts, this is numerically equal to pooling all 1,100 rollout outcomes, while retaining the task-level breakdown. The clean-to-randomized result measures policy performance under the tested shift after clean-demonstration fine-tuning.

A.5 Real-World Policy Experiments

Platform and training data. We use the AgileX COBOT MAGIC implementation of Mobile ALOHA with one overhead camera and two wrist cameras. All three task groups use the right arm. Behavior targets concatenate the overhead and right-wrist streams, giving 640 dimensions. All policy variants use $\pi_{0.5}$ with full-parameter fine-tuning. Comparisons include the base policy, image supervision, V-JEPA2.1 supervision, SigLIP2 (DROID-FT) supervision, and behavior embedding supervision. Table 14 records demonstration composition and evaluation conditions. One placement policy per supervision variant is jointly trained on the two seen instructions.

Table 14: Real-world training data and evaluation conditions. Every reported condition uses 40 trials per supervision variant and a two-minute limit. Unseen pairings have no demonstrations in experimental fine-tuning.

Task group	Fine-tuning demonstrations	Evaluation conditions
Object placement	"put the cup inside the drawer." (56); "put the cube inside the container." (56). Total: 112, jointly trained.	Seen: cup–drawer, cube–container. Unseen: "put the cup inside the container.", "put the cube inside the drawer."
Sequential nesting	Blue setting: 24; red setting: 24. Total: 48.	Standard, Background, Color
Diagonal cloth folding	Pink cloth: 25; blue cloth: 25. Total: 50.	Standard, Color, Texture (patterned cloth)

Instruction compositions and placement success. The held-out instructions recombine familiar object and destination terms: cup–container and cube–drawer are absent from placement fine-tuning, while cup–drawer and cube–container are included. Success requires the requested object in the requested destination. Drawer placement additionally requires closure after placement. The drawer manipulation in the task includes opening while maintaining gripper contact, placing the object, and closing the drawer. Scoring uses the requested final state: additional actions do not invalidate an otherwise successful trial. The seen/unseen comparison therefore tests execution of new instruction compositions alongside the physical demands of the respective destinations.

Sequential nesting and visual changes. The robot first inserts a thin doll into a closely fitting small cup, then places the assembled pair into a larger cup. Full-task success requires the doll inside the small cup and that assembly inside the larger cup. The tight fit makes alignment during insertion

[5] RoboTwin2.0 leaderboard; selection-reference snapshot retained on September 11, 2026.

[6] The RoboTwin2.0 randomized configuration and configuration guide specify these fields.

essential. Evaluation changes either the background beneath the objects or the doll and cup colors, alongside the standard scene. Placement and nesting receive a success count when the requested state is attained within the two-minute trial limit.

Cloth-folding distance. The robot folds diagonally from an upper corner toward the opposite lower corner. Evaluation uses the standard solid-color cloth, a changed color, and a patterned cloth. At the end of each trial, we measure the distance in centimeters between the corner to be folded and its target corner, then average over all 40 trials in that condition. The same measurement includes incomplete executions and trials with no robot motion. For +Image and +V-JEPA2.1 on patterned cloth, no-motion failures leave the cloth unfolded, giving the reported 42.4 cm error: its original diagonal distance under the same metric.

Deployment and reported outcomes. Trials stop at the two-minute limit for every task. The target generators and auxiliary heads are absent from action inference. Table 2 reports all placement and nesting success counts and folding distances for the five policy variants, including SigLIP2 (DROID-FT). Each success count $n/40$ corresponds to $100n/40$ percent; a single trial changes that rate by 2.5 percentage points. Folding distances remain in centimeters and are not combined with the binary task scores.

Observed policy failures. Under the cube–drawer instruction, observed failures include placing the cup in the drawer and closing it, or placing the cube in the container with or without first opening the drawer. These executions attain the wrong object–destination relation. Misalignment during the first doll-to-cup insertion is another observed failure. These observations describe failure types; the aggregate task results do not provide their individual frequencies.

A.6 Behavior Representation Analysis

A.6.1 Features, readouts, and training ablations

Table 3 groups normalized-time, skill, and completion readouts while separating within-task evaluation from held-out-task transfer. Table 18 collects the spatial readouts, and Figure 5 shows qualitative examples.

We analyze frozen embeddings from the DROID-trained encoder on ALOHA demonstrations from RoboTwin2.0, using the recorded `demo_clean_100_aloha` dataset. At each timestep, embeddings from three synchronized camera views are concatenated. Each view contains a 256-dimensional scene-configuration channel s_t and a 64-dimensional motion channel c_t, giving 960 dimensions for the full embedding, 768 for scene configuration, and 192 for motion across the three views. The spatial study evaluates all three input groups; the normalized-time, skill, and completion studies use the full embedding. Representation parameters remain frozen during all readout fitting.

The normalized-time, skill, and completion studies use MLP readouts with two hidden layers of 256 and 128 units and ReLU activations. Inputs are standardized using statistics fitted only on the training portion of each fold. The normalized-time regressor is optimized with Adam using a learning rate of 10^{-3}, weight decay 10^{-4}, and batch size 256, with a maximum of 500 iterations and early stopping. The skill and completion classifiers use the same hidden-layer widths and Adam learning rate, weight decay, and batch size, and are trained for at most 200 epochs. Ten percent of each classifier training fold is reserved for validation; early stopping uses patience 15, and the checkpoint with the lowest validation loss is retained. The spatial readout uses a separate architecture described in Appendix A.6.5.

Unless otherwise stated, the non-spatial RoboTwin probes use fixed episode-level splits with seed 42. Full, w/o End, and w/o Status use the same episode assignments and fitting protocol; only the frozen representation changes. Reported standard deviations, when present, are across folds rather than across independently seeded runs.

Full uses the complete representation-learning objective. The w/o End variant sets $\lambda_{\text{end}} = 0$, removing endpoint reconstruction and its corresponding decoder, while w/o Status sets $\lambda_{\text{status}} = 0$, removing completion and temporal-order supervision together with the corresponding status decoder. Apart from these objective-specific removals, all three encoders use the same DROID training data,

data construction, optimization settings, and checkpoint-selection protocol. In each case, we use the checkpoint saved after exactly one epoch of DROID training, consistent with the training protocol in Appendix A.1.4.

The encoders are then frozen for representation extraction. The ablations use the same extraction and alignment procedure as Full, including the preprocessing, temporal-window construction, and episode-boundary handling described in Appendices A.1.1 and A.3.1. Selecting a channel group from a trained encoder is distinct from removing a training objective during encoder training.

A.6.2 Normalized demonstration time

The normalized-time readout predicts

$$y_i = \frac{i}{L_\eta - 1}, \qquad i = 0, \ldots, L_\eta - 1,$$

where L_η is the number of exported transition embeddings for episode η, and i is their zero-based index. Thus, the targets are linearly spaced from 0 to 1 over the episode's embedding sequence; L_η is distinct from the encoder's fixed window length $T = 60$. The probe receives only the frozen representation.

For each of seven tasks, 100 episodes are divided into ten fixed episode-level folds. Each fold uses 90 episodes for fitting and 10 for evaluation. The readout is an MLP regressor with two hidden layers of 256 and 128 units and ReLU activations. Features are standardized using statistics fitted only on the corresponding training fold. The regressor uses Adam with learning rate 10^{-3}, weight decay 10^{-4}, and batch size 256, with a maximum of 500 iterations and early stopping.

The probe retains all exported transition embeddings without further resampling or removing boundary timesteps. It measures how readily normalized episode position can be recovered from the features used for policy supervision. These features can encode visual evolution together with the temporal context. Full and the ablations share the same extraction protocol, and we interpret this temporal readout alongside the complementary skill, grasp-completion, and spatial probes.

R^2 is computed independently on each held-out fold and then averaged over the ten folds. The normalized-time columns of Table 3a report these mean R^2 values. Full achieves the highest normalized-time score on all seven tasks.

A.6.3 Stage-derived skills and transfer

RoboTwin demonstrations contain structured manipulation phases that provide task-specific skill annotations. The within-task skill study uses the recorded fine-grained phase labels for each task. Each readout is evaluated with episode-level five-fold cross-validation over 100 episodes, giving 80 training and 20 held-out episodes per fold. Macro-F1 is computed independently on each fold and averaged across folds. To reduce class imbalance during MLP fitting, samples from minority classes are upsampled within the training fold to match the largest class.

Task-native within-task labels. The within-task label inventory is task specific and is therefore distinct from the shared vocabulary used for cross-task transfer. Table 15 summarizes the final task-native skill inventories used for the within-task classification results in Table 3a. These are the fine-grained labels used by the reported probe runs; no cross-task remapping is applied in this evaluation.

For six of the seven tasks, the phase-to-skill mapping follows a fixed `phase_id` template shared across all 100 demonstrations. Dump Bin Bigbin is the exception: its fine-grained annotations contain two episode-level trajectory structures rather than a single global phase template. Consequently, its probe labels are generated from the annotations of each individual episode rather than from one fixed phase-index mapping.

For Stack Blocks Three, the task-native annotations distinguish six skills,

$$\{\text{approach},\ \text{grasp},\ \text{lift},\ \text{place},\ \text{release},\ \text{retract}\}.$$

The three repeated manipulation cycles use the same phase-to-skill structure. In the first cycle, event phases 0–3, 4–6, 7–8, 9–12, 13–15, and 16–17 correspond to approach, grasp, lift, place, release, and

Table 15: Task-native skill inventories used for the within-task skill probes on RoboTwin2.0. Each task preserves its own fine-grained annotation vocabulary. These labels are used for the results in Table 3a and are distinct from the shared five-class vocabulary used for cross-task transfer.

Task	Task-native skill inventory
Beat Block Hammer	approach, grasp, lift, transition, strike
Dump Bin Bigbin	approach, grasp, lift, transition, place, release, pour
Handover Mic	approach, grasp, lift, handover, hold
Hanging Mug	approach, grasp, lift_up, transition, lift_down, put_down, release, hang
Lift Pot	preclose, approach, grasp, lift
Put Object Cabinet	approach, grasp, open_door, transition, release
Stack Blocks Three	approach, grasp, lift, place, release, retract

retract, respectively; the same pattern repeats over phases 18–35 and 36–53 for the second and third cycles.

Shared vocabulary for cross-task transfer. The task-native labels above are used only for the within-task skill study. For cross-task evaluation on Blocks Ranking RGB, Blocks Ranking Size, and Stack Blocks Three, the annotations are separately remapped to the common five-class vocabulary

$$\{\text{approach},\ \text{grasp},\ \text{transport},\ \text{release},\ \text{retract}\}.$$

Reach and approach are mapped to *approach*; transition, lift, and place are mapped to *transport*; grasp, release, and retract retain their corresponding labels. Thus, the task-native *lift* and *place* classes of Stack Blocks Three are merged into the coarser *transport* class for cross-task evaluation. The same shared vocabulary is used for all three block-manipulation tasks. Adjacent annotated segments do not overlap: the first frame after the end of one segment belongs to the following segment.

After remapping, the frame counts (approach / grasp / transport / release / retract) are $12045/3000/19717/3000/11518$ for Blocks Ranking RGB, $12055/3000/19826/3000/11526$ for Blocks Ranking Size, and $15527/6600/16865/6600/4680$ for Stack Blocks Three.

For Table 16, the first three rows provide within-task references under this shared five-class vocabulary. Transfer readouts are fitted on Blocks Ranking RGB, Stack Blocks Three, or their pooled union and evaluated directly on Blocks Ranking Size without target-task fitting. RGB $\rightarrow$ Size and Stack $\rightarrow$ Size each use 100 source episodes and 100 target episodes. The $\{$RGB, Stack$\}$ setting concatenates 200 source episodes without additional task-level balancing. Blocks Ranking Size is excluded from fitting, feature standardization, validation, and checkpoint selection; the validation subset used for early stopping is drawn only from the source training data.

The Stack Blocks Three within-task result in Table 16 therefore differs from the task-native within-task result in Table 3a. The former uses the shared five-class vocabulary, whereas the latter preserves the finer six-class task-native vocabulary in which lift and place remain separate. The two values consequently correspond to different label inventories and should not be interpreted as results under an identical classification protocol.

Figure 6 illustrates transfer from the pooled $\{$RGB, Stack$\}$ source set to a held-out Blocks Ranking Size demonstration. The upper timeline shows the shared ground-truth skill sequence, and the lower plot shows the predicted class probabilities at the same embedding timesteps. Table 16 reports aggregate performance for this setting and the two single-source transfer settings.

A.6.4 Grasp completion within and across tasks

The completion probe measures whether a grasp event has reached its completion point. Rather than labeling the full episode, the protocol retains a local window around each grasp cycle containing the approach/reach/preclose portion, the grasp segment, and 30 frames following the grasp. Frames outside this window are excluded.

Let t_{complete} denote the final frame of the annotated grasp segment. Retained frames with $t < t_{\text{complete}}$ are labeled incomplete (0), whereas frames with $t \geq t_{\text{complete}}$ are labeled complete (1).

Table 16: Skill classification within and across RoboTwin2.0 block-manipulation tasks (macro-F1; higher is better). All rows use the shared five-class vocabulary {approach, grasp, transport, release, retract}. Within-task rows use five-fold episode-level evaluation; transfer rows fit on the named source task(s) and evaluate directly on Blocks Ranking Size without target-task fitting. RGB, Size, and Stack denote Blocks Ranking RGB, Blocks Ranking Size, and Stack Blocks Three. Mean averages all six settings, including the within-task references; it is not a transfer-only mean. Bold marks the best value in each row.

Setting	Full	w/o End	w/o Status
Blocks Ranking RGB (within)	**0.740**	0.707	0.739
Blocks Ranking Size (within)	**0.724**	0.691	0.703
Stack Blocks Three (within)	0.794	0.768	**0.803**
RGB → Size	**0.593**	0.540	0.519
Stack → Size	**0.392**	0.386	0.242
{RGB, Stack} → Size	**0.602**	0.566	0.506
Mean (all settings)	**0.641**	0.610	0.585

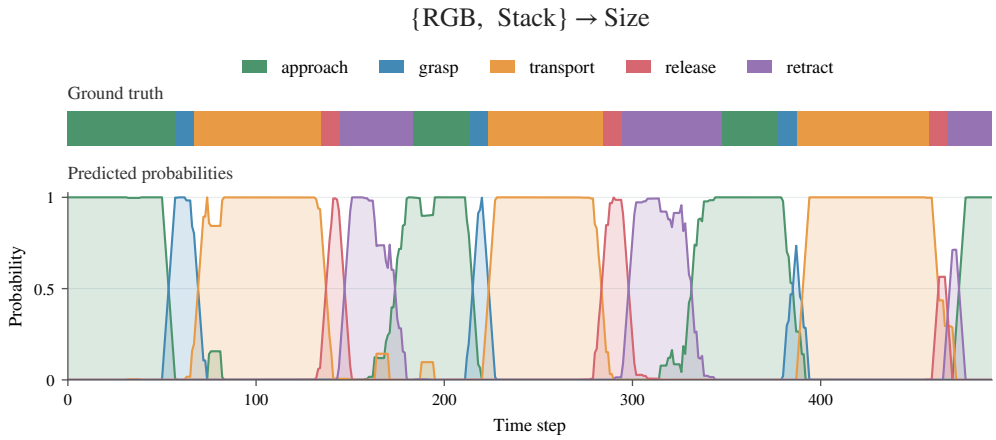


Figure 6: **Cross-task skill predictions on a held-out Blocks Ranking Size demonstration.** The readout is fitted on the pooled {RGB, Stack} source set. The upper timeline shows ground-truth skills; the lower plot shows predicted class probabilities. Colors denote the shared five-class vocabulary: *approach*, *grasp*, *transport*, *release*, and *retract*. Aggregate results appear in Table 16.

Completion is therefore defined at the end of the grasp segment and does not require a subsequent lift stage.

Within-task evaluation uses episode-level five-fold cross-validation with seed 42. Each fold uses 80 episodes for fitting and 20 held-out episodes for evaluation. The frozen representation is standardized using statistics fitted only on the corresponding training data.

Cross-task evaluation follows a leave-one-task-out protocol over the seven completion tasks. For each target task, the readout is fitted on the other six tasks and evaluated directly on the held-out target. With 100 episodes per task, each transfer setting therefore uses 600 source episodes and 100 held-out target episodes. Source-task episodes are pooled without additional task-level balancing. Target-task samples are excluded from feature standardization, validation, early stopping, and checkpoint selection.

For the within-task results, the reported per-task macro-F1 is computed from the pooled held-out predictions across folds. For cross-task evaluation, macro-F1 is computed on the complete held-out target prediction set. Means across tasks give equal weight to the seven held-out task scores.

Across the seven held-out tasks, Full gives the highest macro-F1 on five, while w/o End is higher on Stack Blocks Three and Handover Mic. Averaging the seven reported task-level scores gives mean macro-F1 values of 0.743, 0.714, and 0.704 for Full, w/o End, and w/o Status, respectively, consistent with Table 3b.

The Beat Block Hammer completion panel in Figure 5 uses the Full representation and the leave-one-task-out MLP fitted on the other six tasks. Keyframes show the gripper far from the object, near it, and after lifting it within the retained event window. This probability is the supervised probe output, distinct from the signed status-head score $u_{t,j}$ used during representation learning.

Table 17: Within-task grasp-completion readouts on RoboTwin2.0 (macro-F1; higher is better). Evaluation retains the approach/reach/preclose portion, the grasp segment, and 30 following frames around each grasp event. Frames before the final grasp frame are labeled incomplete (0), while the final grasp frame and subsequent retained frames are labeled complete (1). Readouts are fitted and evaluated on different episodes of each task. Mean weights the seven tasks equally; bold marks the best value in each row.

Task	Full	w/o End	w/o Status
Blocks Ranking RGB	**0.930**	0.912	0.920
Blocks Ranking Size	**0.916**	0.900	0.896
Stack Blocks Three	0.929	0.918	**0.932**
Handover Mic	**0.974**	0.965	0.969
Hanging Mug	**0.949**	0.920	0.943
Beat Block Hammer	**0.962**	0.952	0.951
Put Object Cabinet	**0.958**	0.949	0.955
Mean	**0.945**	0.931	0.938

A.6.5 Skill, block, and spatial-target decoding

We further analyze spatial information on Stack Blocks Three using 100 successful demonstrations. At each retained timestep, the readout predicts (i) the coarse manipulation skill, grasp versus place, (ii) the identity of the manipulated block, (iii) the current active-arm end-effector position (Curr.), and (iv) a skill-conditioned spatial target (Tgt.).

The coarse skill labels are obtained from gripper open/close segmentation rather than from the motion-planning phase annotations used by the preceding skill study. Grasp denotes acquisition of the selected block, while place denotes the subsequent placement phase at the destination. Frames not assigned to either segment are excluded. Block identity follows the red–green–blue manipulation order recorded by the annotation procedure.

Curr. is the 3D active-arm end-effector move-group position in the simulator world frame. Following the recorded probe convention, a fixed $-0.13\,\mathrm{m}$ offset is applied along the world z-axis. This quantity is distinct from the gripper-center TCP. All reported spatial errors are measured in metres.

Tgt. is a stage-constant end-effector keypoint derived automatically from gripper open/close transitions, rather than from object poses, planner waypoints, or manually annotated keypoints. During grasp, the target is the active-arm end-effector position at the beginning of the detected closed gripper segment. During place, the target is the end-effector position at the first open frame following that segment. The same $-0.13\,\mathrm{m}$ world-z offset is applied to the target coordinates.

Features, current positions, and targets are standardized independently using per-dimension means and standard deviations computed only from the training episodes of each fold. The same training statistics are applied to the corresponding validation fold. Spatial predictions are transformed back to the original coordinate scale before computing the reported errors.

The spatial readout first applies a learned input projection $D \to 512$, followed by two 512-unit hidden layers with ReLU and Dropout(0.1). Separate heads predict the binary grasp/place label, the three-way block identity, the 3D current position, and the 3D target position. The optimization

objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}}^{\mathrm{skill}} + \mathcal{L}_{\mathrm{CE}}^{\mathrm{block}} + \mathcal{L}_{\mathrm{MSE}}^{\mathrm{curr}} + 2\,\mathcal{L}_{\mathrm{MSE}}^{\mathrm{tgt}},$$

with the regression losses evaluated in standardized coordinates. Optimization uses AdamW with learning rate 10^{-3}, weight decay 10^{-4}, batch size 512, and 25 epochs. The retained epoch minimizes

$$\mathrm{TgtRMSE} - 0.1\,(\mathrm{SkillAcc} + \mathrm{BlockAcc})$$

on the validation fold.

The spatial protocol uses ten episode-level splits with seed 0. In each split, 90 episodes fit the readout and the remaining ten form the validation fold used for both epoch selection and the reported metrics. The same splits are reused across Full, w/o End, and w/o Status and across All, Scene, and Motion inputs. Skill and block accuracies and spatial errors are averaged over the ten validation folds. These scores report validation performance of the selected checkpoints. Because the same validation episodes select the epoch and supply the reported metrics, performance estimates may be optimistic; no independent test set is used.

For prediction error $e_{i,d}$ at sample i and coordinate $d \in \{x, y, z\}$, the reported errors are

$$\mathrm{RMSE} = \sqrt{\frac{1}{3n}\sum_{i=1}^{n}\sum_{d=1}^{3} e_{i,d}^2}, \qquad \mathrm{MAE} = \frac{1}{3n}\sum_{i=1}^{n}\sum_{d=1}^{3} |e_{i,d}|.$$

Thus, RMSE is computed over all sample-coordinate pairs rather than by first forming a 3D Euclidean distance for each sample.

Table 18 reports the full-embedding and channel-wise results. All concatenates the three 320-dimensional view representations for 960 dimensions. Scene concatenates the 256-dimensional scene channel from each view for 768 dimensions, while Motion concatenates the 64-dimensional motion channel from each view for 192 dimensions. For this spatial readout, the fixed concatenation order is left wrist, right wrist, and head; the policy-target order is given separately in Table 8. This fixed permutation is shared across spatial-probe variants and does not change their feature content.

For the full embedding, target RMSE is 69.5–71.4% lower and current-position RMSE is 76.7–77.2% lower than in the corresponding ablations, while block recognition remains comparable. Full also gives lower target RMSE than both ablations for every input group. Both scene configuration and motion channels therefore contain decodable target information; these results do not establish disjoint semantic roles for the two channel groups.

The middle and right panels of Figure 5 show blue-block grasp and place examples, respectively, on Stack Blocks Three. Markers show the predicted current end-effector position and skill-conditioned spatial target projected into the head-camera view. The grasp target marks block acquisition, while the place target marks the placement destination, using the gripper-transition keypoints defined above. Quantitative errors are reported in Table 18.

The official RoboTwin task descriptions and expert scripts motivate the task-specific interpretations in the main text. Hanging Mug reorients and sets down the mug before the opposite arm picks it up and hangs it; Put Object Cabinet coordinates object handling with opening the drawer. Dump Bin Bigbin includes pickup, transfer where needed, and repeated pouring motions; Lift Pot coordinates both arms before lifting. Stack Blocks Three repeats grasping and transport while the placement surface rises with the stack.[7] These task structures motivate interpretations of the aggregate ablation effects; stage-specific confusion patterns are not measured here.

A.6.6 Action information in the motion channels

On LIBERO-10 demonstrations, we compare the action information accessible from our motion channels with UniVLA latent-action codes (Bu et al., 2025). For our representation, the 64-dimensional motion channels from two camera views are concatenated into a 128-dimensional input. For the UniVLA readout, each camera view supplies four discrete latent-action codes, giving eight codes

[7] Official task descriptions and demonstration videos: Hanging Mug, Put Object Cabinet, Dump Bin Bigbin, Lift Pot, and Stack Blocks Three. The official cabinet expert script grasps the object before opening the drawer and lifting the object.

Table 18: Skill, block, and spatial-target decoding on Stack Blocks Three. All, Scene, and Motion denote the full three-view representation (960D), scene-configuration channels (768D), and motion channels (192D), respectively. The readout uses a learned $D \to 512$ input projection followed by two 512-unit hidden layers with ReLU and Dropout(0.1). Skill labels are grasp/place and block labels identify the manipulated red, green, or blue block. Curr. and Tgt. denote the current active-arm end-effector position and the skill-conditioned grasp or place target in the simulator world frame, including the fixed probe z-offset of $-0.13\,\mathrm{m}$. Classification accuracies and spatial errors are averaged over the ten episode-level validation folds; spatial errors are reported in metres. Each validation fold is also used for epoch selection. Bold marks the best value in each column within each channel setting.

Variant	Channels (dim.)	Skill Acc.	Block Acc.	Curr. RMSE	Tgt. RMSE	Tgt. MAE
Full	All (960)	**0.992**	**1.000**	**0.0153**	**0.0152**	**0.0071**
Full	Scene (768)	**0.990**	**1.000**	**0.0163**	**0.0160**	**0.0075**
Full	Motion (192)	**0.983**	**0.999**	**0.0215**	**0.0230**	**0.0119**
w/o End	All (960)	0.962	0.997	0.0672	0.0532	0.0229
w/o End	Scene (768)	0.960	0.996	0.0691	0.0547	0.0240
w/o End	Motion (192)	0.958	0.997	0.0665	0.0541	0.0237
w/o Status	All (960)	0.985	0.998	0.0656	0.0499	0.0182
w/o Status	Scene (768)	0.980	0.998	0.0669	0.0509	0.0188
w/o Status	Motion (192)	0.973	0.997	0.0654	0.0523	0.0228

across the two views. Each code index $q \in \{0, \ldots, |\mathcal{C}| - 1\}$, where $|\mathcal{C}| = 16$ is the codebook vocabulary size, is mapped to 16 dimensions using the sinusoidal encoding:

$$\phi(q)_{2j} = \sin\left(q/10000^{2j/16}\right), \qquad \phi(q)_{2j+1} = \cos\left(q/10000^{2j/16}\right), \qquad j = 0, \ldots, 7.$$

The eight 16-dimensional encodings are concatenated into a 128-dimensional input, preserving code order within each view. Both readouts use dimensions $128 \to 256 \to d_{\text{out}}$.

For each Cartesian axis $d \in \{x, y, z\}$, let $\Delta p_{\eta,t,d} = p_{\eta,t+1,d} - p_{\eta,t,d}$ denote the next-minus-current end-effector position change in demonstration η. We define a fixed dataset-level stillness threshold

$$\tau_d = 10^{-1} \max_{\eta,t} |\Delta p_{\eta,t,d}|,$$

computed once over all LIBERO-10 demonstrations. A transition is labeled zero on axis d when $|\Delta p_{\eta,t,d}| \le \tau_d$, and negative or positive according to its sign otherwise. Gripper labels are taken directly from LIBERO's binary open/close signal.

Both representations use fixed episode-level ten-fold outer splits. Each readout is trained for 20 epochs, with the checkpoint selected using a validation split within the outer training portion. The selected checkpoint is evaluated on the separate held-out fold, which is not used for checkpoint selection.

A.7 Qualitative Subprocess Segmentation

We visualize subprocess partitions in demonstrations of two real-world policy tasks: cup placement and cloth folding. The figures illustrate the greedy motion-coherence rule in Eq. 1 at different tolerances ϵ over each complete episode. Figures 7 and 8 show selected partitions, using sampled frames to illustrate their temporal structure.

The selected partitions reveal recognizable phases of the demonstrated interactions. For cup placement, the finest displayed partition separates approach to the drawer, drawer opening, cup grasping, cup transport and placement, and drawer closing. For cloth folding, it distinguishes approach, grasping the cloth corner, folding, and withdrawal. The coarser displayed partitions group several of these phases into longer subprocesses, illustrating different temporal extents under the same coherence criterion.

Table 19: Action-label classification on LIBERO-10 using our motion channels and UniVLA latent-action codes. Ours concatenates two 64-dimensional view features; UniVLA concatenates eight 16-dimensional sinusoidal code encodings (four codes per view) into a 128-dimensional input. MLPs have dimensions $128 \to 256 \to d_{\text{out}}$. Direction labels are negative/zero/positive for each Cartesian axis; gripper labels are binary open/close. Results are evaluated on held-out folds of fixed episode-level ten-fold splits, with checkpoints selected on separate validation data. Acc., Prec., Rec. and F1 denote accuracy, precision, recall and F1; all are higher-is-better.

			Macro Avg.			Weighted Avg.	
Model	Action	Acc. ↑	Prec. ↑	Rec. ↑	F1 ↑	Prec. ↑	F1 ↑
Ours	x	0.6313	0.5990	0.5943	0.5898	0.6726	0.6471
Ours	y	0.7097	0.6858	0.6882	0.6794	0.7476	0.7229
Ours	z	0.7308	0.6624	0.6888	0.6637	0.7801	0.7484
Ours	gripper	0.9569	0.9563	0.9571	0.9566	0.9571	0.9569
UniVLA	x	0.5926	0.5612	0.5563	0.5527	0.6302	0.6071
UniVLA	y	0.7091	0.7045	0.6983	0.6852	0.7717	0.7281
UniVLA	z	0.7043	0.6404	0.6583	0.6355	0.7640	0.7263
UniVLA	gripper	0.8258	0.8244	0.8250	0.8246	0.8262	0.8259

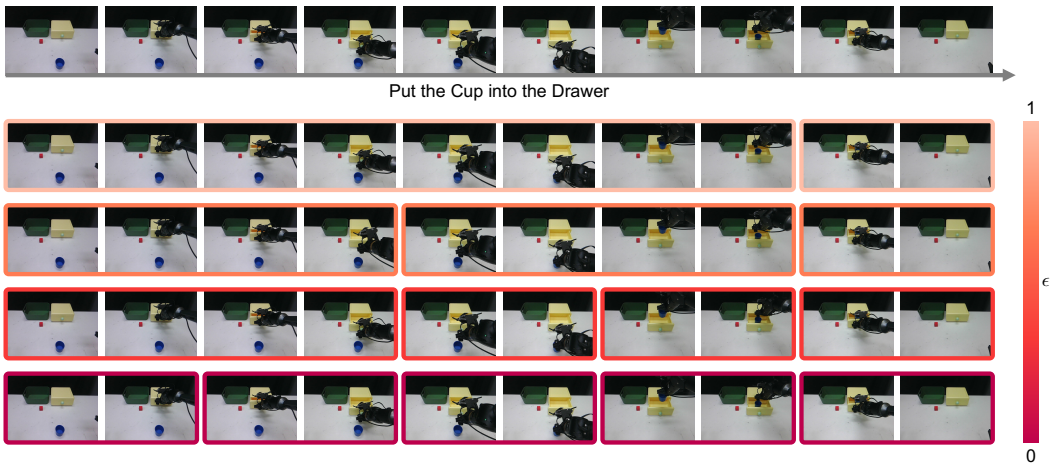


Figure 7: **Subprocess partitions for cup placement.** The top strip shows a demonstration of putting a cup into a drawer; the rows below show four selected partitions, from higher to lower tolerance ϵ. Outlines group sampled frames from the same subprocess; colors indicate relative tolerance ordering.

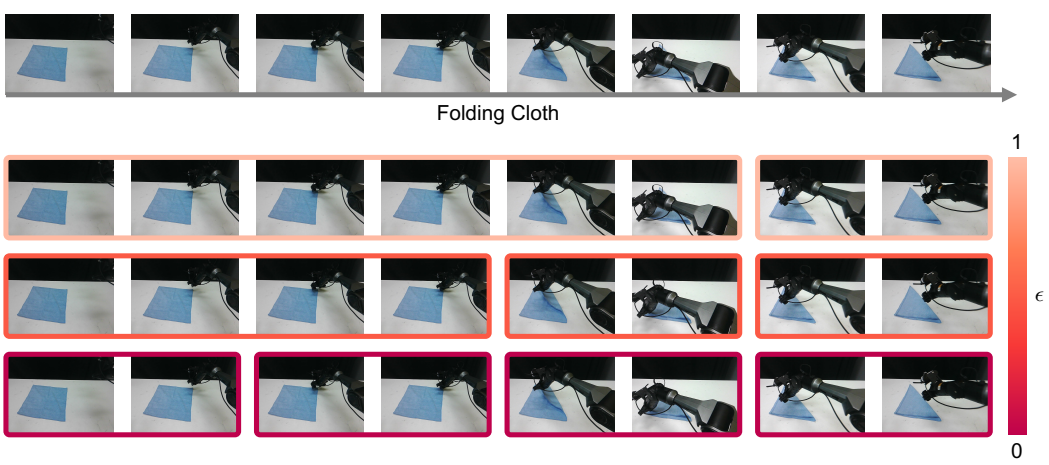


Figure 8: **Subprocess partitions for cloth folding.** The top strip shows a cloth-folding demonstration; the rows below show three selected partitions, from higher to lower tolerance ϵ. Outlines group sampled frames from the same subprocess; colors indicate relative tolerance ordering.

A.8 Additional Results and Ablations

Policy result coverage. The main tables report the complete simulation task matrices and real-world results. Table 20 summarizes their measurement protocols; the summaries below are computed from the reported task-level values.

Table 20: Policy result locations and measurement protocols.

Study	Main result location	Measurement
Simulation policies	Table 1	100 rollouts per variant, task and condition; mean over 11 tasks
Auxiliary targets	Table 1	$\pi_{0.5}$ LoRA; 100 clean rollouts per target and task
Placement and nesting	Table 2	Goal-state successes / 40 trials; two-minute limit
Cloth folding	Table 2	Mean final corner distance over all 40 trials; two-minute limit

Gains across policies and evaluation conditions. Table 21 summarizes the within-policy comparison between the native objective and the addition of behavior supervision. Every clean task improves for each of the three policies. Under randomized evaluation, clean-trained $\pi_{0.5}$ improves on nine tasks and ties Base on Place Dual Shoes and Stamp Seal. Its higher randomized success does not imply a smaller clean-to-randomized drop: the drop is 24.5 percentage points for behavior supervision and 19.0 for Base, computed from the unrounded means.

Table 21: Descriptive summary of the simulation policy comparisons. Success averages 11 tasks equally, with 100 rollouts per task. Gains are calculated before rounding. Improved/tied/worse counts compare task-level success rates.

Policy	Evaluation	Base (%)	Ours (%)	Gain (pp)	Improved/ tied/worse
π_0	Clean	20.8	31.4	10.5	11/0/0
FastWAM	Clean	51.0	65.4	14.4	11/0/0
$\pi_{0.5}$	Clean	37.2	50.2	13.0	11/0/0
$\pi_{0.5}$	Randomized	18.2	25.6	7.5	9/2/0

Table 22: V-JEPA auxiliary-target variants with the same base-policy and behavior-embedding references as Table 1. Scores are success rates (%) over 100 rollouts per task; Mean weights the 11 tasks equally. Pool uses the extraction in Appendix A.3.4; PCA jointly projects the three views to 960 dimensions using only policy-training demonstrations. Target dim. counts all three views. Bold marks the largest reported value among the displayed rows, including ties.

Auxiliary target	Target dim.	Blocks Ranking Size	Stack Blocks Three	Scan Object	Place Mouse Pad	Place Dual Shoes	Place Bread Basket	Stamp Seal	Place Object Basket	Hanging Mug	Place Object Scale	Blocks Ranking RGB	Mean
None (Base)	–	25.0	17.0	23.0	24.0	11.0	79.0	48.0	47.0	8.0	70.0	57.0	37.2
Ours	960	**55.0**	**30.0**	**41.0**	**38.0**	**17.0**	**86.0**	70.0	55.0	**15.0**	**74.0**	71.0	**50.2**
V-JEPA2.1 pool	2304	34.0	11.0	37.0	32.0	12.0	76.0	62.0	**67.0**	3.0	65.0	**80.0**	43.5
V-JEPA2.1 pool + DROID-FT	2304	28.0	16.0	36.0	30.0	10.0	77.0	70.0	24.0	7.0	72.0	74.0	40.4
V-JEPA2.1 PCA + DROID-FT	960	54.0	10.0	25.0	31.0	**17.0**	81.0	**74.0**	13.0	2.0	63.0	77.0	40.6

Seen and held-out instruction compositions. Pooling the two seen placement conditions gives 70 successes in 80 trials (87.5%) for behavior supervision, compared with 56/80 (70.0%) for Base, 64/80 (80.0%) for +Image, 67/80 (83.75%) for V-JEPA2.1, and 55/80 (68.75%) for SigLIP2 (DROID-FT). For the two held-out compositions, the corresponding counts are 49/80 (61.25%), 0/80, 14/80 (17.5%), 0/80, and 40/80 (50.0%). These aggregates weight the two instructions equally; the individual conditions in Table 2 show the complementary result that SigLIP2 leads on cup–container while behavior supervision leads on cube–drawer. Nesting and folding are reported separately for each appearance condition because they test different changes and use different outcome measures.

V-JEPA extraction variants. Table 22 compares an additional PCA-projected V-JEPA2.1 target alongside the pooled V-JEPA2.1 targets used in the main comparison. All rows use $\pi_{0.5}$ LoRA on the same 11 clean tasks. Pool denotes spatial pooling followed by selection of the first encoded temporal position from a 64-frame clip; concatenating three views yields 2304 coordinates. The additional DROID-FT PCA target has 960 coordinates in total, equal to the three-view behavior-target width used in RoboTwin. It uses the same DROID-fine-tuned V-JEPA2.1 checkpoint: the three 768-dimensional view features are first concatenated, then jointly projected from 2,304 to 960 dimensions by PCA fitted only on policy-training demonstrations. The fitted projection is kept fixed when generating targets. This tests reducing target width for the same DROID-adapted encoder and auxiliary MSE prediction procedure.

Probe ablations. Table 23 summarizes the encoder-training ablations used throughout the representation analyses. Both ablations retain next-frame prediction. Apart from removing the corresponding objective and decoder, the three encoders use the same DROID training data, optimization settings, and checkpoint-selection protocol, with the checkpoint saved after exactly one epoch of DROID training. The same representation-extraction procedure is used across variants. Study-specific readout architectures, episode splits, annotations, and metric aggregation are reported in Appendix A.6.

Table 23: Encoder-training ablations used in the representation analyses. Apart from the specified objective and decoder removal, encoder training and representation extraction are shared across variants. Probe-specific fitting and evaluation protocols are given in Appendix A.6.

Variant	Encoder-training objective
Full	Complete embedding-learning objective.
w/o End	Remove endpoint reconstruction and its corresponding decoder ($\lambda_{\text{end}} = 0$).
w/o Status	Remove completion and temporal-order supervision together with the corresponding status decoder ($\lambda_{\text{status}} = 0$).

Policy evidence and complementary supervision. Behavior embeddings achieve the highest reported mean in the evaluated auxiliary-target comparison, including the additional 960-dimensional PCA target, while other representations lead some tasks. The comparison supports the value of

the learned behavior targets under the stated policy interface; the extraction and upstream learning configurations are detailed in Appendix A.3.4. The paired policy studies establish added value across the three tested policy configurations. FastWAM's improvement while retaining dense WAN prediction further shows that endpoint- and status-grounded behavior supervision can be useful alongside an existing predictive objective. Real-world evaluations extend this utility to the tested instruction compositions and appearance changes in precise and deformable manipulation.

Representation and generalization scope. Offline probes measure accessibility under their specified features, temporal context and labels. They do not alone show what shared policy features contain or which cues a policy uses at inference. Skill and normalized-time labels offer limited human-interpretable references; they do not validate a complete semantic hierarchy, calibrated completion or general failure recognition. Real-world language results cover the reported object–destination recombinations, and visual results cover the specified scene/appearance changes. These studies support the observed conditions and motivate broader investigation without establishing every form of language or visual generalization.

Qualitative examples and failure cases. Figures 6 and 5 illustrate supervised skill, completion, and spatial readouts, while Figures 7 and 8 show subprocess partitions obtained directly from motion-channel coherence. These visualizations serve different purposes: probe predictions assess information accessible from frozen embeddings, and the partitions illustrate the discovery rule at different tolerances. Observed policy errors, including incorrect object–destination pairings and nesting misalignment, are described in Appendix A.5; their aggregate outcomes are reported in Table 2.

A.9 PSEUDOCODE FOR EMBEDDING LEARNING.

Algorithms 1 and 2 describe subprocess discovery and one training iteration for a window of $T \geq 2$ frames. Transition t connects frames t and $t+1$; a subprocess $[a, g)$ contains transitions $a, \ldots, g-1$ and ends at frame g. Multi-horizon supervision is obtained by resampling one tolerance per window on each iteration. The encoder does not receive the tolerance. We write $[v]_+ = \max(0, v)$, $\mathrm{normalize}_2(v) = v/\|v\|_2$, and sg for stop-gradient. For unit motion channels, $\mathrm{dist}(c_p, c_q) = \pi^{-1} \arccos\big(\mathrm{clip}(\langle c_p, c_q \rangle, -1, 1)\big)$.

Algorithm 1 Greedy multi-horizon subprocess partitioning

Require: Unit motion channels $c_1, \ldots, c_{T-1}$; tolerance $\epsilon \in [0, 1]$
Ensure: Ordered partition $\mathcal{P}$ of transitions $1, \ldots, T-1$

function PARTITION($c_{1:T-1}, \epsilon$)
  $D_{pq} \leftarrow \mathrm{dist}(c_p, c_q)$ for all $p, q \in \{1, \ldots, T-1\}$
  $\mathcal{P} \leftarrow ()$, $a \leftarrow 1$
  while $a < T$ **do**
    $g \leftarrow a + 1$ ▷ A singleton transition is coherent
    while $g < T$ **do**
      if $\max_{a \leq j < g} D_{jg} > \epsilon$ **then**
        break ▷ Adding transition g violates the diameter bound
      end if
      $g \leftarrow g + 1$
    end while
    Append $[a, g)$ to $\mathcal{P}$
    $a \leftarrow g$
  end while
  return $\mathcal{P}$
end function

The lowercase losses denote one stochastic realization of the corresponding expected losses in the main text. MSE averages all grid entries, including retained cells in endpoint reconstruction. Endpoint corruption is redrawn for every transition, with $\rho = 0.85$. The order loss compares each ongoing frame using its own scene configuration channel and the shared prototype; it is zero at the final transition of each subprocess. All task losses are averaged over $T-1$ transitions, including zero

Algorithm 2 One training iteration for behavior embeddings from one window

Require: Frames $\mathbf{o}_{1:T}$; frozen image encoder Φ_{DC}; parameters $\Theta = \{\Theta_{\mathcal{E}}, \Theta_{\mathcal{F}}, \Theta_{\mathcal{G}}, \Theta_{\mathcal{U}}\}$
Ensure: Updated encoder and task-network parameters Θ

$x_t \leftarrow \Phi_{\mathrm{DC}}(\mathbf{o}_t), t = 1, \dots, T$ ▷ Frozen grids; may be cached
$(z_1, \dots, z_{T-1}) \leftarrow \mathcal{E}(x_{1:T}; \Theta_{\mathcal{E}})$, with $z_t = [s_t; c_t]$
    Both channels are unit-normalized by the encoder.
$\hat{c}_t \leftarrow \mathrm{sg}(c_t), t = 1, \dots, T-1$
$d_j \leftarrow \max_{1 \le p,q \le j} \mathrm{dist}(\hat{c}_p, \hat{c}_q), j = 1, \dots, T-1$
$(b_0, \dots, b_{q+1}) \leftarrow \mathrm{sort}\big(\{0,1\} \cup \{d_j : 0 < d_j < 1\}\big)$
    Here q counts distinct interior cut points; $b_0 = 0$ and $b_{q+1} = 1$.
Draw $J \sim \mathrm{Unif}\{0, \dots, q\}$, then $\epsilon \sim \mathrm{Unif}[b_J, b_{J+1})$
$\mathcal{P} \leftarrow \textsc{Partition}(\hat{c}_{1:T-1}, \epsilon)$ ▷ Keep tolerance and boundaries fixed
for each subprocess $I_k = [a, g)$ in $\mathcal{P}$ **do**
    Draw independent $r_j \sim \mathrm{Unif}(0,1), j \in I_k$
    $\bar{c}_k \leftarrow \mathrm{normalize}_2\big(\sum_{j \in I_k} r_j c_j\big)$ ▷ Shared prototype; retain gradients
    for $t = a, \dots, g-1$ **do**
        $\tilde{z}_t \leftarrow [s_t; \bar{c}_k]$
        $\ell^{(t)}_{\mathrm{next}} \leftarrow \mathrm{MSE}\big(\mathcal{F}(x_t, \bar{c}_k, \epsilon; \Theta_{\mathcal{F}}), x_{t+1}\big)$
        For each spatial cell i, independently draw $m_{t,i} \sim \mathrm{Bernoulli}(\rho)$, $\eta_{t,i} \sim \mathcal{N}(0, I_{d_x})$
        $x^{\mathrm{noisy},(t)}_{g,i} \leftarrow (1 - m_{t,i}) x_{g,i} + m_{t,i} \eta_{t,i}$ for every cell i
        $\ell^{(t)}_{\mathrm{end}} \leftarrow \mathrm{MSE}\big(\mathcal{G}(x^{\mathrm{noisy},(t)}_g, \tilde{z}_t, \epsilon; \Theta_{\mathcal{G}}), x_g\big)$
        $u_{t,t} \leftarrow \mathcal{U}(x_t, \tilde{z}_t, \epsilon; \Theta_{\mathcal{U}})$, $u_{t,g} \leftarrow \mathcal{U}(x_g, \tilde{z}_t, \epsilon; \Theta_{\mathcal{U}})$
        $\ell^{(t)}_{\mathrm{comp}} \leftarrow [\delta_{\mathrm{cur}} + u_{t,t}]_+ + [\delta_{\mathrm{end}} - u_{t,g}]_+$
        $\ell^{(t)}_{\mathrm{order}} \leftarrow 0$
    end for
    for $t = a, \dots, g-2$ **do** ▷ Empty for a singleton subprocess
        $\ell^{(t)}_{\mathrm{order}} \leftarrow [\delta_{\mathrm{order}} - (u_{t+1,t+1} - u_{t,t})]_+$
    end for
end for
$C \leftarrow [c_1^\top; \dots; c_{T-1}^\top]$
$\widehat{R}(C) \leftarrow \frac{1}{2} \log\det\Big(I_{d_c} + \frac{d_c}{(T-1)\sigma^2_{\mathrm{reg}}} C^\top C\Big)$
$\mathcal{L}_{\mathrm{emb}} \leftarrow \dfrac{1}{T-1} \displaystyle\sum_{t=1}^{T-1} \Big[\lambda_{\mathrm{next}} \ell^{(t)}_{\mathrm{next}} + \lambda_{\mathrm{end}} \ell^{(t)}_{\mathrm{end}} + \lambda_{\mathrm{status}}\big(\ell^{(t)}_{\mathrm{comp}} + \ell^{(t)}_{\mathrm{order}}\big)\Big] - \lambda_{\mathrm{reg}} \widehat{R}(C)$
Update Θ with one AdamW step using $\nabla_\Theta \mathcal{L}_{\mathrm{emb}}$.

order terms and padding. For this single-window presentation, C contains only that window's motion channels; the training implementation pools motion channels across the local minibatch for coding-rate regularization. Sampling and partitioning receive no gradient, whereas prototype construction and coding rate use the original, differentiable motion channels. Optimizer state and the learning-rate schedule are omitted for brevity. The remaining settings are $(\lambda_{\mathrm{next}}, \lambda_{\mathrm{end}}, \lambda_{\mathrm{status}}) = (1, 1, 1)$, $\lambda_{\mathrm{reg}} = 0.01$, $(\delta_{\mathrm{cur}}, \delta_{\mathrm{end}}, \delta_{\mathrm{order}}) = (1, 1, 0.5)$, and $\sigma^2_{\mathrm{reg}} = 0.05$.