EACT: COUPLING STRUCTURED EFFECTS MANIFOLD AND ACTION TRAJECTORIES FOR ROBOT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

World-action models guide robot control by anticipating how a scene will change. However, similar visual transitions can arise from trajectories with different motion paths, orientation changes, and gripper timing. Learning to reconstruct future observations therefore does not ensure that the resulting representation captures how the change is carried out. We introduce **EAct**, which couples a structured effect space with a trajectory-supervised action representation. Its geometry-regularized effect space encourages transitions with similar visual changes to have nearby effect representations. Differences between frozen current and future visual features define the similarity reference, adding relational supervision beyond reconstruction. An effect-conditioned ActionVAE learns trajectory-level execution representations from demonstrations. Through effect-guided policy learning, the policy predicts an effect, decodes it into future visual features, and uses dense transition features to predict an action latent that conditions a flow-matching expert. EAct achieves average success rates of 99.0% across the four LIBERO task suites and 93.04% under RoboTwin clean evaluation, outperforming the compared baselines on both simulation benchmarks. On LIBERO-Object, an effect-only variant achieves comparable success to the original-latent baseline with 80% fewer Stage-2 updates. Code will be released.

1 INTRODUCTION

World Action Models (WAMs) incorporate future prediction into policy learning, enabling action generation to exploit anticipated scene changes. One line of work predicts the future in observation space, where the resulting frames are decoded into actions, used as subgoals, or generated jointly with the actions (Du et al., 2023; Wu et al., 2024; Hu et al., 2025; Bi et al., 2026; Cen et al., 2025; Guo et al., 2026; Kim et al., 2026). This instantiation can be computationally expensive when iterative denoising over high-dimensional video latents is repeatedly required during control. Feature-space WAMs instead move prediction into a compact visual-feature space: they learn a low-dimensional *transition representation* that summarizes the change from the current observation to a future observation, and expose it to the action generator, thereby retaining an explicit future at a fraction of the cost of video generation (Chen et al., 2026a; Lyu et al., 2026; Zhang et al., 2026; Liu et al., 2026a; Hung et al., 2026; Huang et al., 2026). These transition representations are supervised by future observation features or by reconstruction residuals, and are therefore viewed as a description of *what happened* in the scene—the effect of an interaction.

However, even when starting from similar initial states and ending at similar endpoints, trajectories can differ in their motion paths, their orientation changes, and the timing of their gripper operations. Therefore, the map from transition to execution is one-to-many (Figure 1), and the resulting ambiguity can impair action reconstruction, motivating the incorporation of trajectory information into these representations.

Learning a trajectory-level representation is not itself the main challenge: prior work has long compressed demonstrations into plan or skill latents (Lynch et al., 2020; Pertsch et al., 2021), while recent methods learn continuous trajectory latents jointly with policies (Yang et al., 2026) or optimize trajectory latents at test time (Wang et al., 2026) or constrain such latents using both trajectories and future observations (Xu et al., 2026). Nevertheless, simply adding an action representation to a feature-space WAM introduces two difficulties. First, the action representation can re-encode 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.

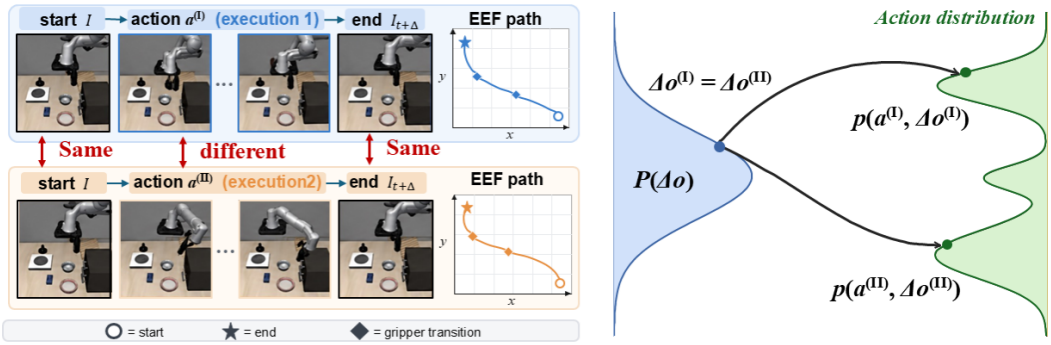

Figure 1: **Endpoint-similar, execution-different windows.** (a) Two executions share the same current and future observations while differing in their intermediate motions. (b) A single effect distribution $p(z_e \mid I_t, I_{t+\Delta})$ can be compatible with multiple action distributions $p(z_a \mid z_e, s_t)$. In the image labels, $o_t = I_t$ and $o_{t+K} = I_{t+\Delta}$; K counts the control intervals spanning Δ.

scene change that the effect representation already carries, so the two require an explicit division of labor—the action side should explain the residual trajectory variation left once the effect is given. Second, the complete action sequence is available only at training time, so the residual execution information must be recovered at deployment from the current observation, the predicted effect, and the anticipated future.

We introduce **EAct**, which decouples the transition representation into a geometry-regularized *effect representation* z_{e} and a trajectory-supervised *action representation* z_{a}. The effect representation is learned from current–future observations to describe observable scene changes, but future reconstruction alone does not constrain the neighborhood relations among different transitions. We therefore align the local neighborhoods of the effect space with a reference given by differences between frozen current and future visual features, so that similar transitions remain neighbors in both spaces. The action representation is instead produced by an effect-conditioned ActionVAE trained on complete demonstration sequences. At inference time, the policy predicts an effect, decodes it into future visual features, and infers the action representation from task semantics, the current state, and the dense visual transitions between current and predicted features. The predicted future features and the action representation then jointly condition a flow-matching action expert. At deployment, the policy uses current observations, robot state, and the instruction, without access to ground-truth future observations or demonstrated trajectories.

We evaluate EAct on LIBERO (Liu et al., 2023) and RoboTwin (Chen et al., 2026b), and analyze both the neighborhood structure of its effect space and the execution information encoded by its action representation. EAct achieves an average success rate of 99.0% across the four LIBERO task suites and 93.0% under the RoboTwin clean setting. Representation analyses show that neighborhood regularization improves local agreement with the visual-transition reference, and that trajectory probes recover more execution information from the action representation than from prior transition representations.

Our contributions are threefold:

- **Geometry-regularized effect representations.** We augment future-feature reconstruction with neighborhood alignment based on frozen visual-feature differences, using transition-similarity supervision to organize the local geometry of the effect space.
- **Effect-conditioned action representations and policy learning.** We learn an action representation from complete demonstrated trajectories and use it as an intermediate policy prediction target, connecting effect prediction, anticipated visual transitions, execution prediction, and action generation.
- **Control evaluation and representation analysis.** We demonstrate the effectiveness of EAct on LIBERO and RoboTwin, and examine the distinct roles of its two representations through neighborhood-preservation analysis and trajectory-execution probing.

2 Related Works

World Action Model. World action models (WAMs) introduce environment dynamics prediction into robot policies, enabling action generation to exploit anticipated futures (Du et al., 2023; Wu et al., 2024; Hu et al., 2025; Bi et al., 2026; Cen et al., 2025; Guo et al., 2026). One family of WAMs keeps future prediction and action generation within a single generative stream, operating either in pixel space or in the generative latent space of a video model (Bi et al., 2026; Cen et al., 2025; Guo et al., 2026; Kim et al., 2026). These models provide explicit predictions of future observations, but can incur the cost of iterative generation during action inference and retain visual details irrelevant to control. A complementary line of work retains future prediction only as a training-time objective or an internal-layer constraint, removing the need for explicit future generation at inference (Yuan et al., 2026; Zheng et al., 2025; Zhao et al., 2026). Feature-space world action models reduce the burden of visual prediction by operating in a compact visual-feature space rather than reconstructing pixels (Lyu et al., 2026; Zhang et al., 2026; Liu et al., 2026a; Hung et al., 2026). LaWAM further trains a policy to predict a representation of an observation transition and decode it into future patches that condition an action expert (Chen et al., 2026a). Building on this efficient feature-space prediction framework, we structure the transition representation as a geometry-regularized *effect representation* and a trajectory-supervised *action representation*, so that both the intended change and the execution process that realizes it guide control.

Latent Action Model. Latent action models (LAMs) encode observation transitions from unlabeled videos into latent actions that serve as pseudo-action supervision for policy pretraining. Existing approaches typically combine inverse dynamics modeling with future reconstruction to capture transition information (Schmidt & Jiang, 2024; Ye et al., 2025), while more action-centric representations are encouraged through disentanglement of task-relevant factors, cross-view reconstruction, structured latent constraints, or alignment with action-policy features (Bu et al., 2025; Lee et al., 2026; Wei et al., 2026; Tang et al., 2026; Li et al., 2026b; Liu et al., 2026b). Beyond visual transitions, latent representations can also be learned directly from executable trajectories by compressing action segments into skills or trajectory plans (Pertsch et al., 2021; Lynch et al., 2020; Jiang et al., 2023), using temporal autoencoding or jointly modeling action reconstruction and environment-conditioned visual consequences (Wang et al., 2026; Xu et al., 2026). Our framework treats visual transitions and action trajectories as complementary interfaces: the effect latent z_{e} captures observable scene changes and is organized by the neighborhood geometry of visual transitions, whereas the action latent z_{a} is supervised by ground-truth trajectories rather than induced from unlabeled video. The policy distills both latents and uses them to generate executable action chunks.

3 Methods

3.1 Overall Architecture

Given the current observation I_t, robot state s_t, and language instruction ℓ, our goal is to generate an action chunk $A_t \in \mathbb{R}^{H\times d_a}$, where H is the number of control commands and d_a is the dimension of each command. The chunk spans a physical interval Δ. A frozen visual encoder maps the current and future observations to F_t and $F_{t+\Delta}$. Stage 1 learns an effect representation of observable scene change and an action representation of its execution from future observations and demonstrated trajectories. Stage 2 trains the model to predict these representations from current information to condition a flow-matching action expert (Figure 2).

Stage 1: learning effect and action representations. An *effect encoder* takes F_t and $F_{t+\Delta}$ and encodes their visual transition into an effect latent z_e. Following LaWAM (Chen et al., 2026a), we use *latent world model* (LaWM), denoted by W_ω, to combine F_t with z_e to reconstruct the future features. This reconstruction teaches the effect latent to describe scene change. We further apply neighborhood regularization to encourage similar visual changes to have nearby effect codes.

We then fix the effect encoder and train the *effect-conditioned ActionVAE*. Its action encoder compresses the demonstrated chunk A_t into an action latent z_a, and its action decoder reconstructs the trajectory. Both are conditioned on the current state s_t and the effect encoder's posterior mean $z_e^\star$, so that z_a captures execution information given the effect. After training, the two encoders are frozen as

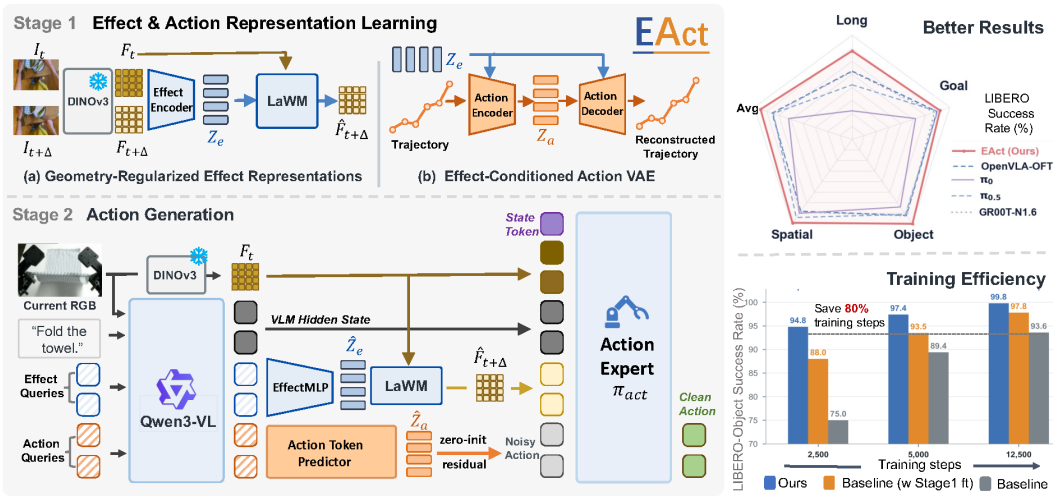

Figure 2: **Overview of our effect–action architecture. Stage 1** learns (a) the effect representation, then (b) the ActionVAE with the effect encoder fixed; z_e, z_a are reconstruction samples. **Stage 2** trains the policy by distilling both frozen Stage 1 teachers' frozen targets $z_e^\star$, $q_a^\star$ to predict $\widehat{z}_e$, future features $\widehat{F}_{t+\Delta}$, and an action distribution whose mean $\widehat{z}_a$ conditions the expert.

teachers: their effect mean $z_e^\star$ and action posterior $q_a^\star$ supervise Stage 2. Representation learning is detailed in Section 3.2.

Stage 2: representation distillation and policy learning. We train the policy by distilling both frozen Stage 1 teachers while learning action generation from demonstrations. Qwen3-VL processes the current image and instruction to provide task context. The *Effect MLP* predicts an effect $\widehat{z}_e$, which LaWM combines with current visual features to anticipate the future. The *Action Latent Predictor* uses this anticipated transition, together with the current state and task context, to infer an execution representation p_θ^a. The predicted future features and execution representation then condition the flow-matching *Action Expert* to generate an action $\widehat{A}_t$. The policy follows

$$\begin{aligned} \widehat{z}_e &= \pi_e(I_t, \ell), & \widehat{F}_{t+\Delta} &= W_\omega(F_t, \widehat{z}_e), \\ p_\theta^a &= \pi_a(I_t, \ell, F_t, \widehat{F}_{t+\Delta}, s_t, \widehat{z}_e), & \widehat{A}_t &= \pi_{\text{act}}(I_t, s_t, \ell, \widehat{F}_{t+\Delta}, \widehat{z}_a). \end{aligned} \tag{1}$$

where $\widehat{z}_a = \mathbb{E}_{p_\theta^a}[z_a]$. The π interfaces include VLM processing and suppress fixed embodiment and horizon arguments. Future observations and demonstrations provide training targets; deployment requires only I_t, s_t, ℓ. Section 3.3 details the distillation objectives and policy training.

3.2 Effect-Conditioned Action Representations

In **Stage 1**, we first train the effect encoder and world decoder W_ω jointly, then fix both and train the effect-conditioned ActionVAE. Both posterior encoders are frozen before Stage 2.

Effect Learning. Accurate future reconstruction does not ensure that similar visual changes have nearby effect latents. The effect encoder observes the current and future features, and the world decoder reconstructs the future from the current features and a posterior sample:

$$z_e \sim q_{\phi_e}(\cdot \mid F_t, F_{t+\Delta}), \qquad \widetilde{F}_{t+\Delta} = W_\omega(F_t, z_e). \tag{2}$$

To organize the effect space, we use differences between frozen visual features as a reference for transition similarity. For each transition i, we construct two neighborhood distributions over the same set of comparable transitions, excluding i itself. The visual reference distribution p_i^V is computed from visual-transition distances, while p_i^E is computed from distances between effect posterior means.

In both spaces, closer transitions receive higher probability. We train the effect encoder to match the visual reference:

$$\mathcal{L}_{\text{nbr}} = \mathbb{E}_i\big[D_{\text{KL}}\big(\text{sg}(p_i^V) \,\|\, p_i^E\big)\big]\,. \tag{3}$$

The expectation averages anchors with valid candidate sets. Descriptor construction, affinities, and candidate sampling are detailed in Appendix A.2. Effect learning combines this alignment with future-feature reconstruction, state-change prediction, and variational regularization. The full objective is given in Appendix A.3.

ActionVAE. The ActionVAE encodes complete demonstrated chunks A_t, conditioned on the current state and frozen effect mean, $c_t = [s_t; z_e^\star]$:

$$z_a \sim q_{\phi_a}(\cdot \mid A_t, c_t), \qquad \widetilde{A}_t = D_{\psi_a}(z_a, c_t). \tag{4}$$

Reconstruction supervises motion and gripper commands, with variational regularization on the posterior. We then freeze the action encoder and decoder, which provide the teacher targets

$$z_e^\star = \text{sg}(\mu_e(F_t, F_{t+\Delta})), \qquad q_a^\star = \text{sg}(q_{\phi_a}(\cdot \mid A_t, s_t, z_e^\star)). \tag{5}$$

Here $z_e^\star$ is the effect posterior mean, $q_a^\star$ is the action posterior distribution, and $z_a^\star = \mathbb{E}_{q_a^\star}[z_a]$ is its mean. In contrast, z_e, z_a in the reconstruction equations denote posterior samples. The operator sg stops gradients through teacher targets. Stage 2 matches the effect mean and action posterior, retaining the frozen ActionVAE decoder for auxiliary trajectory supervision. Appendix A.3 gives the objectives and posterior parameterization.

3.3 Effect-Guided Policy Learning

Stage 2 combines distillation from the frozen Stage 1 teachers with action-policy training. Qwen3-VL (Bai et al., 2025) processes I_t, ℓ with learnable effect and plan queries Q_e, Q_{plan}, producing output features H_e, H_{plan} for the two prediction branches, as well as a set of visual-language tokens.

Effect distillation. The effect head g_e (the Effect MLP in Figure 2) maps H_e to the predicted effect, which the world decoder uses to predict future features:

$$\widehat{z}_e = g_e(H_e), \qquad \widehat{F}_{t+\Delta} = W_\omega(F_t, \widehat{z}_e). \tag{6}$$

The effect distillation loss $\mathcal{L}_e$ matches $\widehat{z}_e$ to the frozen teacher mean $z_e^\star$, using the fixed normalization in Appendix A.4. The future-feature loss $\mathcal{L}_f$ matches $\widehat{F}_{t+\Delta}$ to the observed features $F_{t+\Delta}$.

Action distillation and conditioning. A dense transition predictor fuses current features, predicted future features, and their difference at each patch location into spatial transition tokens. The VLM output H_{plan}, conditioned on s_t and $\widehat{z}_e$, cross-attends to these tokens to predict an action-latent distribution, whose mean is used for action generation:

$$p_\theta^a = g_a(H_{\text{plan}}, F_t, \widehat{F}_{t+\Delta}, s_t, \widehat{z}_e), \qquad \widehat{z}_a = \mathbb{E}_{p_\theta^a}[z_a]. \tag{7}$$

Thus instruction information enters g_a through H_{plan}. The action distillation loss $\mathcal{L}_a$ matches p_θ^a to the frozen posterior $q_a^\star$ through posterior matching and mean regression toward $z_a^\star$. It also includes auxiliary trajectory reconstruction through the frozen ActionVAE decoder. Distribution and loss details are in Appendix A.6.

A learned adapter maps the predicted mean $\widehat{z}_a$ to an additive residual on the noisy-action embeddings, which encode the noisy action chunk and flow time. The expert also receives vision-language tokens from Qwen3-VL and projections of s_t, F_t, and $\widehat{F}_{t+\Delta}$ (Figure 2). Predictor and adapter details are given in Appendices A.5 and A.7.

Joint policy training. We combine both distillation objectives with conditional flow matching (Lipman et al., 2023) and future-feature supervision:

$$\mathcal{L}_{\text{policy}} = \mathcal{L}_{\text{FM}} + \underbrace{\lambda_e \mathcal{L}_e + \lambda_a \mathcal{L}_a}_{\text{distillation of Stage 1 representations}} + \lambda_f \mathcal{L}_f, \tag{8}$$

The flow-matching loss $\mathcal{L}_{\text{FM}}$ trains action generation from noisy demonstration chunks. Appendices A.6 and D give the loss definitions, weights, and conditioning curriculum that progressively enables flow-loss gradients to reach the action predictor. At inference, the expert generates an action chunk from noise while holding the predicted context and action latent fixed.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmark and Evaluation. We evaluate on LIBERO (Liu et al., 2023) and RoboTwin 2.0 (Chen et al., 2026b). LIBERO comprises four evaluation suites—Spatial, Object, Goal, and Long (LIBERO-10)—with ten tasks per suite and 50 rollouts per task using the benchmark-provided initial states. RoboTwin evaluation covers 50 bimanual tasks, with 100 rollouts per task under each of the clean and randomized conditions. We report task-averaged success rates, averaging the four suite scores for LIBERO and reporting the two conditions separately for RoboTwin. We compare against VLA policies, latent-action-based methods, and world-action models (Tables 1 and 2), with LaWAM (Chen et al., 2026a) as the primary architectural baseline.

Implementation Details. We use Qwen3-VL-2B-Instruct with frozen DINOv3 ViT-B/16 features and 32-dimensional effect and action latents. Each benchmark uses one multitask policy initialized from its released LaWAM SFT checkpoint. LIBERO finetuning uses Original demonstrations for 25k additional updates with global batch 256; For RoboTwin tasks, we use the merged RoboTwin demonstrations for 50k additional updates with global batch 1,024. Posterior teachers are trained in stage1 and remain frozen during policy training. Policy chunks contain eight actions at 20 Hz on LIBERO and 36 actions at 30 Hz on RoboTwin. Optimizer settings, phase schedules, and checkpoint selection are detailed in Appendix D.

4.2 ROBOT MANIPULATION PERFORMANCE

LIBERO. EAct reaches a 99.0% average and matches or improves upon LaWAM on every suite (Table 1). The largest gain occurs on Long, while performance on Goal is preserved and the already high Object and Spatial scores improve further. Moreover, extending LaWAM training by 25,000 updates does not reproduce the gain, suggesting that it does not arise from additional optimization alone.

Method	Long	Goal	Object	Spatial	Average
MAINSTREAM VLA					
OpenVLA-OFT (Kim et al., 2025)	94.5	97.9	98.4	97.6	97.1
π_0 (Black et al., 2025b)	88.4	94.4	96.8	98.0	94.4
$\pi_{0.5}$ (Black et al., 2025a)	92.4	98.0	98.2	98.8	96.9
GR00T-N1.6 (NVIDIA GEAR Team, 2025)	94.4	97.5	98.5	97.7	97.0
LATENT ACTION BASED METHODS					
LAPA (Ye et al., 2025)	55.4	58.8	74.6	73.8	65.7
UniVLA (Bu et al., 2025)	92.0	95.6	96.8	96.5	95.2
Mantis (Yang et al., 2025)	94.2	94.4	99.2	98.8	96.7
VLA-JEPA (Sun et al., 2026)	95.8	97.2	99.6	96.2	97.2
MAINSTREAM WAM					
F1 (Lv et al., 2025)	91.3	95.4	97.8	98.2	95.7
Motus (Bi et al., 2026)	<u>97.6</u>	96.6	<u>99.8</u>	96.8	97.7
Cosmos-Policy (Kim et al., 2026)	<u>97.6</u>	<u>98.2</u>	**100.0**	98.1	98.5
LingBot-VA (Li et al., 2026a)	**98.5**	97.2	99.6	98.5	98.5
Fast-WAM (Yuan et al., 2026)	95.2	97.0	**100.0**	98.2	97.6
LaWAM (Chen et al., 2026a)	97.0	**98.4**	99.6	<u>99.4</u>	<u>98.6</u>
LaWAM (+25k steps) (Chen et al., 2026a)	96.6	97.0	<u>99.8</u>	99.2	98.2
EAct	<u>97.6</u>	**98.4**	**100.0**	**99.8**	**99.0**

Table 1: **Success rates (%) on the four LIBERO task suites.** EAct achieves an average success rate of 99.0%, including 100.0% on LIBERO-Object. LaWAM (+25k steps) denotes LaWAM with 25,000 additional training steps. Bold and underlined values denote the best and second-best distinct scores in each column, including ties.

RoboTwin. We test the same design on RoboTwin bimanual tasks. EAct improves the LaWAM average under both clean and randomized evaluation, from 92.64% to 93.04% and from 89.80% to 90.08%, respectively (Table 2). The gains across both conditions indicate that the effect–action

Task	Fast-WAM		GigaWorld-Policy		LingBot-VA		$\pi_{0.5}$		Motus		LaWAM		**EAct**	
	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
Lift Pot	**100**	**100**	98	98	**100**	**100**	96	85	96	99	**100**	99	**100**	**100**
Move Can Pot	95	92	76	78	93	93	51	55	34	74	**98**	93	**98**	96
Move Stapler Pad	84	63	92	82	59	71	56	42	83	85	94	**87**	**97**	82
Pick Diverse Bottles	81	88	82	70	93	87	81	71	90	**91**	91	88	**96**	83
Place Dual Shoes	88	88	96	84	94	81	75	75	93	87	**98**	94	96	**97**
				...	(10 selected tasks; 50 tasks in total)									
Place Object Scale	83	88	88	80	**98**	87	86	80	88	85	95	88	**98**	**91**
Place Shoe	97	98	98	96	99	97	92	93	99	97	**100**	**100**	**100**	**100**
Press Stapler	**98**	**100**	96	96	82	88	87	83	93	98	**98**	97	97	**100**
Rotate QRcode	95	94	90	84	92	**95**	89	87	89	73	94	89	94	**95**
Stack Blocks Two	**100**	**100**	**100**	94	**100**	99	97	**100**	**100**	98	**100**	97	**100**	99
Average (50 tasks)	91.98	90.52	86.36	85.04	91.50	**90.92**	82.74	76.76	88.66	87.02	92.64	89.80	**93.04**	90.08

Table 2: **RoboTwin results on selected tasks.** Success rates (%) under clean (Clean) and randomized (Rand.) settings. Ten selected tasks are shown; averages are computed over all 50 tasks, with full per-task results in the appendix. Bold and underlined values indicate the best and second-best results within each setting, respectively.

factorization shows robustness across datasets. The results therefore support a consistent improvement over the underlying world-action architecture.

4.3 Component Ablations

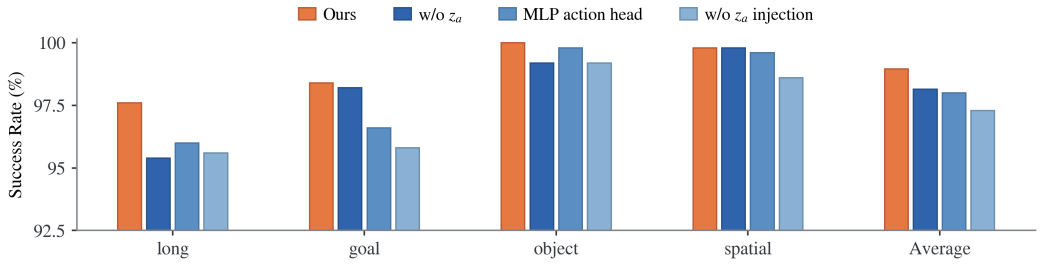

Figure 3: **Component ablations on LIBERO.** Full EAct is compared with action-branch removal, an MLP predictor, a predictor-only future-input intervention, disabled action-latent injection, and the transferred RoboTwin ActionVAE (Section 4.5).

We examine how the action token and its predictor contribute to control (Figure 3). Removing the action-token prediction and conditioning branch reduces average success from 98.95% to 98.15%, supporting the contribution of trajectory-supervised execution information. To assess the token's direct conditioning role, we retain action-latent supervision but disable its residual injection into the expert; success drops to 97.30%. To evaluate the predictor architecture, we replace the cross-attention predictor with an MLP action head while retaining action-latent supervision and injection. Average success falls to 98.00%, with the largest decrease on Goal, from 98.4% to 96.6%; Long also decreases from 97.6% to 96.0%. Together, these comparisons support explicit action-token conditioning and favor cross-attention-based action-latent prediction over the MLP alternative. Protocols and suite-level results are given in Appendix E.3.

4.4 Representation Analysis

Neighborhood Preservation in Effect Latent. To verify whether manifold regularization organizes effect latents by visual-transition similarity, we examine local-neighbor preservation. We evaluate 2,975 non-overlapping transitions from 650 held-out LIBERO episodes across 130 tasks. The effect encoders use identical architecture, data, seed, and 40-epoch schedule and with only manifold constraint different. Overlap@5 is the fraction of five nearest scene-change neighbors also retrieved in effect space.

Metric	w/o manifold	Ours
Overlap@5: within task	61.18	**63.77**
Overlap@5: global	14.39	**18.20**
Overlap@5: context matched	39.10	**43.98**
ρ_{local}	0.372	**0.517**

Table 3: **Matched-budget representation ablation on LIBERO.** *Overlap*@5is reported as a percentage; ρ_{local} denotes within-task Spearman correlation. Higher values indicate better neighborhood preservation.

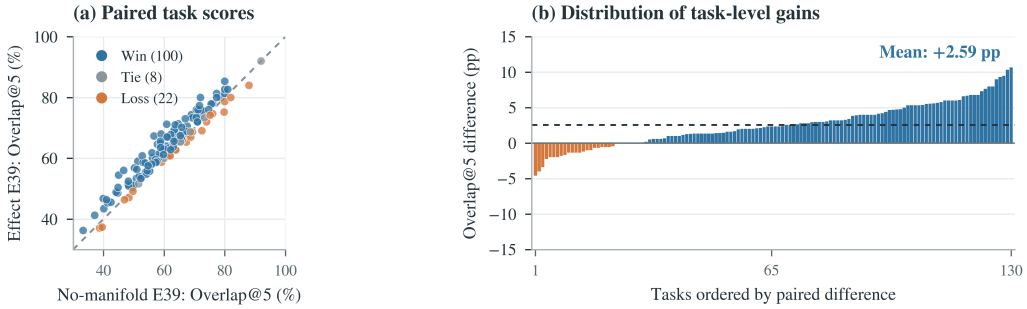

Figure 4: **Task-level consistency of neighborhood preservation.** (a) Paired within-task Overlap@5 for all 130 tasks; the diagonal indicates equal scores. (b) Task-level differences, sorted by value, with the dashed line marking the mean improvement of 2.59 percentage points. Manifold regularization improves neighborhood overlap on 100 of 130 tasks.

Regularization improves within-task Overlap@5 from 61.18% to 63.77% (Table 3), with gains on 100 of 130 tasks (Figure 4). It also improves context-matched overlap from 39.10% to 43.98% and local distance-rank correlation from 0.372 to 0.517. Thus, regularization better preserves DINOv3-defined local transition geometry across tasks and similar starting contexts. Definitions and aggregation details appear in Appendix E.1.

	Ridge		MLP	
Input	Teacher	Predicted	Teacher	Predicted
$\mathbf{s}$	0.0356	0.0356	0.0699 ± 0.0018	0.0699 ± 0.0018
$[\mathbf{s}; \mathbf{z}_e]$	0.1614	0.1649	0.2611 ± 0.0026	0.2641 ± 0.0017
$[\mathbf{s}; \mathbf{z}_{\text{LaWAM}}]$	0.1597	0.1769	0.2910 ± 0.0014	0.3017 ± 0.0020
$[\mathbf{s}; \mathbf{z}_a]$	**0.5570**	**0.3417**	$\mathbf{0.7183} \pm 0.0031$	$\mathbf{0.4275} \pm 0.0029$

Table 4: **Execution-residual task-macro R^2.** Teacher latents encode observed transitions or trajectories; predicted latents are policy-generated. MLP results report mean $\pm$ sample standard deviation over three probe seeds. All state-plus-latent inputs have the same dimensionality.

Action Latent Semantic Probe. To verify whether the action latent captures execution beyond endpoint displacement, we probe frozen representations for intermediate motion. At 20 Hz, a 0.4-s window yields eight waypoints. We subtract the straight position path and shortest orientation interpolation to the endpoint so performance reflects deviations from a direct endpoint path; targets retain residuals and gripper displacements from initial width. Ridge (L2-regularized linear regression) and an MLP predict these 50D targets from state plus one latent. We evaluate 1,941 windows from 16 Stage-2 training episodes across eight tasks, held out from probe fitting; evaluation is in-distribution. Details appear in Appendix E.2.

Action latents perform best for teacher and predicted representations with both probes (Table 4). For predicted latents, Ridge R^2 increases from LaWAM's 0.1769 to 0.3417 on all eight tasks, while MLP R^2 rises from 0.3017 to 0.4275. These matched-width results show that path and gripper information

is more accessible from the action latent than from state, effect, or LaWAM latents, and remains accessible after policy prediction.

4.5 Learning Efficiency and Transfer

Learning efficiency. To assess our effect representation's contribution to policy-learning efficiency, we compare the effect-only variant with the original and Stage-1-finetuned baselines on LIBERO-Object across three Stage-2 checkpoints (Figure 5). At 2,500 updates, it reaches 94.8% success, exceeding the original baseline's 93.6% at 12,500 updates with 80% fewer updates. At the same 12,500-update budget, it achieves 99.8%, compared with 97.8% for the Stage-1-finetuned baseline. These results support our effect representation as guidance for learning successful policies with fewer updates. Representation-training costs are excluded; protocol details are in Appendix E.5.

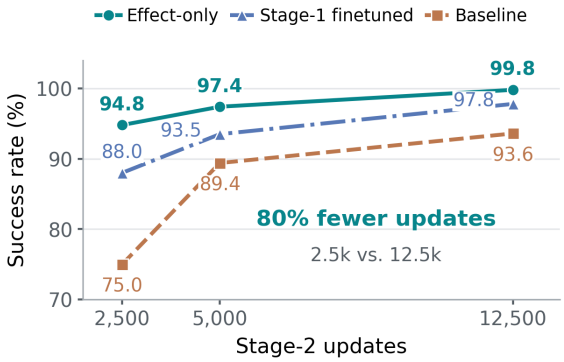

Figure 5: **Stage-2 efficiency on LIBERO-Object.** Checkpoints at 10%, 20%, and 50% of a 25k-update schedule. Stage-1 training costs are excluded.

Cross-dataset transfer. To assess whether the action representation transfers across datasets, we reuse the frozen RoboTwin-trained ActionVAE and train the downstream policy on LIBERO Original, using fixed embodiment and temporal adapters. Without finetuning the teacher, the policy achieves 97.55% average success across the four LIBERO suites, supporting action-teacher reuse with downstream policy adaptation. Protocols and suite-level results are in Appendix E.4.

5 Conclusion

We presented EAct, which couples geometry-regularized effect representations with trajectory-supervised action representations for robot control. Neighborhood alignment organizes the effect space by visual-transition similarity, while an effect-conditioned ActionVAE captures execution information from demonstrations. The policy predicts these representations from current inputs and uses anticipated futures and action latents to guide action generation. EAct achieves 99.0% average success on LIBERO and 93.04% under RoboTwin's clean setting. Representation analyses and component ablations support the complementary roles of effect structure and execution information. The effect-only study further demonstrates faster Stage-2 learning on LIBERO-Object, while cross-dataset experiments show that a frozen action teacher can be reused with downstream policy adaptation. These findings support explicitly modeling both observable outcomes and their execution in world-action policies.

6 Limitations

Our experiments cover simulated manipulation on LIBERO and RoboTwin; performance on physical robots remains unvalidated. The action probes measure short-horizon execution information within the Stage-2 training distribution, and cross-dataset transfer still requires downstream policy training and embodiment and temporal adapters. These results leave broader generalization to new tasks and embodiments open.

The geometry objective aligns effect latents with frozen visual-feature differences, which approximate observable change without directly measuring physical effects. Its utility depends on the visual encoder and the transitions represented in the demonstrations. Training also requires representation learning and teacher-target preparation in addition to policy optimization. The reported efficiency gain concerns effect-only Stage-2 checkpoints on LIBERO-Object and excludes these preparation costs; it does not establish a reduction in total training compute.

AI USE STATEMENT

We used generative AI tools for language editing, identifying related work, and implementing parts of the code. We did not use these tools for idea formation, generating synthetic datasets, or proving mathematical claims. We reviewed all AI-assisted material and manually checked the suggested references and text. Two authors verified and tested the AI-generated code. We take responsibility for the final content of this work, including text, claims, and artifacts produced with AI assistance.

REPRODUCIBILITY STATEMENT

We provide the model architecture and learning objectives in Section 3, with notation, representation objectives, normalization conventions, action prediction, and flow-matching definitions in Appendix A. Appendix B reports the backbone, visual encoder, latent dimensions, action horizons, and inference settings. The representation-training procedure, policy objectives, optimization hyperparameters, phase-specific schedules, and checkpoint selection criteria are documented in Appendix D. Section 4 specifies the LIBERO and RoboTwin evaluation protocols; complete RoboTwin per-task results and the representation-analysis procedures are included in the appendices. We will release the training and evaluation code to support reproduction of the reported results.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 35101–35113, 2026.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.014.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

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

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. arXiv:2506.18088.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023. doi: 10.52202/075280-0403.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346. PMLR, 2025.

Jiakai Huang, Zhongbo Wu, Zheng Zhang, Zihan Wang, Shan You, and Tao Huang. Foresight without seeing: Latent futures for world action models. *arXiv preprint arXiv:2608.11605*, 2026.

Adam Hung, Bardienus P. Duisterhof, Deva Ramanan, and Jeffrey Ichnowski. Modality-autoregressive world-action models. *arXiv preprint arXiv:2609.17524*, 2026.

Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. In *The Eleventh International Conference on Learning Representations*, 2023.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.017. OpenVLA-OFT.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2601.16163.

Jung Min Lee, Dohyeok Lee, Seokhun Ju, Taehyun Cho, Jin Woo Koo, Li Zhao, Sangwoo Hong, and Jungwoo Lee. MVP-LAM: Learning action-centric latent action via cross-viewpoint reconstruction. In *Forty-third International Conference on Machine Learning*, 2026.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026a. LingBot-VA.

Qiwei Li, Xicheng Gong, Xinghang Li, Peiyan Li, Quanyun Zhou, Hangjun Ye, Jiahuan Zhou, and Yadong Mu. RotVLA: Rotational latent action for vision-language-action model. *arXiv preprint arXiv:2605.13403*, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*, 2023. doi: 10.52202/075280-1939.

Jiahao Liu, Zhongpu Xia, Shuai Tian, Huangrui Li, Yuhang Zheng, Ning Ma, Xin Fu, Xiaotian Liu, Jing Li, Yixian Li, ShangQing Zhou, Zebin Xing, Linbo Wang, Chaoyue Li, Haoran Li, and Dongbin Zhao. WALA: Learning executable latent actions from action-labeled demonstrations and action-free videos. *arXiv preprint arXiv:2607.11397*, 2026a.

Mengya Liu, Baoxiong Jia, Jiangyong Huang, Jingze Zhang, and Siyuan Huang. LARA: Latent action representation alignment for vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026b.

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

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 30 Oct–01 Nov 2020.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, Wenbo Cui, Senmao Qi, Shuo Wang, Yixin Zheng, Mi Yan, Xuesong Shi, Haoran Li, Dongbin Zhao, Ming-Yu Liu, Zhizheng Zhang, Li Yi, Yizhou Wang, and He Wang. LDA-1B: Scaling latent dynamics action model via universal embodied data ingestion. In *Proceedings of Robotics: Science and Systems*, 2026. doi: 10.15607/RSS.2026.XXII.210.

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

NVIDIA GEAR Team. GR00T N1.6: An improved open foundation model for generalist humanoid robots. https://research.nvidia.com/labs/gear/gr00t-n1_6/, December 2025. NVIDIA Research technical report. Authors listed alphabetically.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. doi: 10.1609/aaai.v32i1.11671.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 2021. CoRL 2020; arXiv:2010.11944.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *The Twelfth International Conference on Learning Representations*, 2024.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, 2026. Featured Certification.

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 (ECCV)*, 2026.

Zuojin Tang, Haoyun Liu, Xinyuan Chang, Changjie Wu, Dongjie Huo, Yandan Yang, Bin Liu, Zhejia Cai, Feng Xiong, Mu Xu, Jiachen Luo, De Ma, Zhiheng Ma, and Gang Pan. ALAM: Algebraically consistent latent action model for vision-language-action models. *arXiv preprint arXiv:2605.10819*, 2026.

Jiaxu Wang, Yicheng Jiang, Tianlun He, Jingkai Sun, Qiang Zhang, Junhao He, Jiahang Cao, Zesen Gan, Mingyuan Sun, Qiming Shao, and Xiangyu Yue. Mvista-4d: View-consistent 4d world model with test-time action inference for robotic manipulation. In *43rd International Conference on Machine Learning*, 2026.

Hangxing Wei, Xiaoyu Chen, Chuheng Zhang, Tim Pearce, Jianyu Chen, Alex Lamb, Li Zhao, and Jiang Bian. Learning additively compositional latent actions for embodied AI. *arXiv preprint arXiv:2604.03340*, 2026.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuecheng Xu, Tong Yang, Jingkai Jia, Chi Zhang, Xuelong Li, and Wenqiang Zhang. EDAR: Learning environment-dependent action representations for robotic manipulation. *arXiv preprint arXiv:2607.11427*, 2026.

Tong Yang, Jingkai Jia, Yuecheng Xu, Xueyao Chen, Chi Zhang, and Wenqiang Zhang. Trajectory-level continuous action representation for robotic manipulation. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 442–458, Cham, 2026. Springer Nature Switzerland.

Yi Yang, Xueqi Li, Yiyang Chen, Jin Song, Yihan Wang, Zipeng Xiao, Jiadi Su, You Qiaoben, Pengfei Liu, and Zhijie Deng. Mantis: A versatile vision-language-action model with disentangled visual foresight. *arXiv preprint arXiv:2511.16175*, 2025.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025.

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.

Xinyu Zhang, Zhengtong Xu, Yutian Tao, Yeping Wang, Yu She, and Abdeslam Boularias. Learning visual feature-based world models via residual latent action. *arXiv preprint arXiv:2605.07079*, 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 Björck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. FLARE: Robot learning with implicit world modeling. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3952–3971. PMLR, 2025.

A Supplementary Method Details

A.1 Notation

Symbol	Definition
t, Δ	Decision time and physical duration of a chunk.
H, d_a	Number of action samples in a chunk and dimension of each command.
$A_t \in \mathbb{R}^{H \times d_a}$	Normalized action chunk after the dataset's temporal preprocessing.
$I_t, I_{t+\Delta}, s_t, \ell$	Current/future images, current robot state, and instruction.
$F_t, F_{t+\Delta} \in \mathbb{R}^{P \times C}$	Frozen DINOv3 patch features; P positions and C channels.
d_e, d_z	Effect and action latent dimensions, both 32.
q_{ϕ_e}, q_{ϕ_a}	Effect and conditional-action posteriors used for representation learning.
$\mu_e, \sigma_e^2; \mu_a, \sigma_a^2$	Posterior means and diagonal variances, not standard deviations.
z_e, z_a	Latent samples during representation training.
$z_e^\star, q_a^\star, z_a^\star$	Frozen effect mean, action posterior, and its mean $z_a^\star = \mathbb{E}_{q_a^\star}[z_a]$.
$\widetilde{F}, \widetilde{A}$	Reconstructions obtained from posterior samples.
$\mathcal{N}_e, u_e^\star, \widehat{u}_e$	Fixed effect normalization, normalized teacher mean, and predicted normalized code.
$\pi_e, \pi_a, \pi_{\text{act}}$	Complete prediction paths, including VLM processing.
$g_e = \mathcal{N}_e^{-1} \circ f_e$	Effect head on H_e, including restoration to native coordinates.
$\widehat{z}_e$	Predicted effect restored to native world-model coordinates.
$\widehat{F}_{t+\Delta}$	Policy-driven future features, not a ground-truth future.
$p_\theta^a, \widehat{\mu}_a, \widehat{\sigma}_a^2$	Predicted conditional action Gaussian and its parameters.
$\widehat{z}_a = \widehat{\mu}_a$	Predicted Gaussian mean passed to the action expert.
Q_e, Q_{plan}	Learnable effect and plan queries supplied to the VLM.
$H_e, H_{\text{plan}}, H_{\text{sem}}$	VLM output features for effect, plan, and vision-language conditioning.
$T, Q^{(l)}$	Dense transition memory and plan-query states at decoder layer l.
b, b_p^{pos}	Embodiment identifier and spatial positional embedding; unrelated quantities.
R_ξ, v_η	Action-code residual adapter and action velocity network.
$\tau \in [0, 1]$	Flow interpolation time, distinct from the physical horizon Δ.
$\epsilon_e, \epsilon_a, \epsilon$	Independent effect-latent, action-latent, and action-flow Gaussian noise.
sg	Stop-gradient; when applied to a Gaussian, it freezes its parameters.

The image notation suppresses camera indices. A 256×256 image produces a 16×16 grid of DINOv3 patch features. We distinguish the physical chunk duration Δ from the flow interpolation time τ. Teacher posterior means are deterministic targets; latent samples are used during variational representation learning.

A.2 Neighborhood Construction

Visual-transition descriptors. For transition i and patch p, we form a residual between normalized frozen visual features:

$$r_{i,p} = \text{sg}[\mathcal{U}(F_{t+\Delta,i,p}) - \mathcal{U}(F_{t,i,p})]\,,$$
$$\mathcal{U}(x) = \frac{x}{\|x\|_2 + \varepsilon}, \qquad \varepsilon > 0. \tag{9}$$

Current and future images use matching preprocessing and patch ordering; CLS and register tokens are excluded. We project the flattened residual onto a fixed PCA basis $U \in \mathbb{R}^{PC \times d_r}$:

$$v_i = U^\top(\text{vec}(r_i) - m_r)\,,$$
$$s_V = \sqrt{\frac{1}{N_{\text{tr}} d_r} \sum_{i=1}^{N_{\text{tr}}} \|v_i\|_2^2}, \qquad e_i = \frac{v_i}{s_V + \varepsilon}. \tag{10}$$

The mean m_r, PCA basis U, and scalar scale s_V are estimated from training residuals and then frozen. Scalar RMS scaling preserves relative variation across PCA coordinates. The resulting descriptor provides a reference for visual-transition similarity; it is not a calibrated measure of physical change.

Dissimilarities and candidate sets. We compare visual descriptors with a Huber dissimilarity and effect posterior means with a normalized squared Euclidean distance:

$$\rho_1(x) = \begin{cases} \frac{1}{2}x^2, & |x| \le 1, \\ |x| - \frac{1}{2}, & |x| > 1, \end{cases}$$
$$d^V_{ij} = \frac{1}{d_r}\sum_{c=1}^{d_r} \rho_1(e_{i,c} - e_{j,c}), \qquad d^E_{ij} = \frac{\|\mu_{e,i} - \mu_{e,j}\|_2^2}{d_e}. \tag{11}$$

For each anchor i, let $\mathcal{M}_i$ denote its eligible reference pool, excluding self-pairs. The candidate set combines visual and latent nearest neighbors with references at intermediate and large visual dissimilarities:

$$\mathcal{C}_i = \mathcal{N}^V_{k_V}(i) \cup \mathcal{N}^E_{k_E}(i) \cup \mathcal{R}_{\mathrm{mid}}(i) \cup \mathcal{R}_{\mathrm{far}}(i) \subseteq \mathcal{M}_i. \tag{12}$$

Candidates are deduplicated, and the two affinity distributions use the same support. Candidate selection is discrete; the neighborhood objective does not differentiate through the selection operation.

Affinity alignment. Let $h_i = \max(d^V_{i,(k_V)}, \varepsilon)$, where the order statistic is computed in $\mathcal{M}_i$, and let $\gamma > 0$ be a fixed effect-kernel scale. For $j \in \mathcal{C}_i$, define

$$p^V_{ij} = \frac{\exp[-(d^V_{ij})^2/(2h_i^2)]}{\sum_{k\in\mathcal{C}_i}\exp[-(d^V_{ik})^2/(2h_i^2)]},$$
$$p^E_{ij} = \frac{(1 + d^E_{ij}/\gamma^2)^{-1}}{\sum_{k\in\mathcal{C}_i}(1 + d^E_{ik}/\gamma^2)^{-1}}, \tag{13}$$
$$\mathcal{L}_{\mathrm{nbr}} = \frac{1}{|\mathcal{B}_{\mathrm{valid}}|}\sum_{i\in\mathcal{B}_{\mathrm{valid}}}\sum_{j\in\mathcal{C}_i} \mathrm{sg}(p^V_{ij}) \log\frac{\mathrm{sg}(p^V_{ij})}{p^E_{ij}}.$$

Here $\mathcal{B}_{\mathrm{valid}}$ contains anchors for which the candidate set and bandwidth are defined. The visual distribution is fixed with respect to model parameters. Alignment supervises the effect posterior means; it does not directly constrain posterior variances. Since d^E already denotes a squared distance, it enters the Student-t affinity without a second squaring.

A.3 Representation Reconstruction and Priors

Both encoders use diagonal Gaussian posteriors. For $k \in \{e, a\}$, suppressing the conditioning inputs, we write

$$q_{\phi_k} = \mathcal{N}(\mu_k, \mathrm{diag}(\sigma_k^2)), \qquad z_k = \mu_k + \sigma_k \odot \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(0, I). \tag{14}$$

The effect encoder receives $(F_t, F_{t+\Delta})$. Its decoder predicts $\widetilde{F}_{t+\Delta} = W_\omega(F_t, z_e)$, and its objective is

$$\mathcal{L}_{\mathrm{effect}} = \mathbb{E}\Big[\mathrm{SmoothL1}_{0.1}(\widetilde{F}_{t+\Delta}, F_{t+\Delta})\Big] + \lambda_s\mathcal{L}_{\mathrm{state}} + \beta_e\mathbb{E}[D_{\mathrm{KL}}(q_{\phi_e}\,\|\,\mathcal{N}(0, I))] + \lambda_n\mathcal{L}_{\mathrm{nbr}}. \tag{15}$$

The auxiliary state term supervises normalized proprioceptive changes where the corresponding state targets are available. For a residual x, the SmoothL1 convention is

$$\mathrm{smooth}_\delta(x) = \begin{cases} x^2/(2\delta), & |x| < \delta, \\ |x| - \delta/2, & |x| \ge \delta, \end{cases}$$
$$\mathrm{SmoothL1}_\delta(X, Y) = \frac{1}{|X|}\sum_j \mathrm{smooth}_\delta(X_j - Y_j). \tag{16}$$

For a diagonal posterior, the unmodified standard-normal KL is

$$D_{\mathrm{KL}}(q\,\|\,\mathcal{N}(0, I)) = \frac{1}{2}\sum_{j=1}^{d}\left(\mu_j^2 + \sigma_j^2 - 1 - \log\sigma_j^2\right). \tag{17}$$

The ActionVAE is conditioned on $c_t = [s_t; \mathrm{sg}(\mu_e(F_t, F_{t+\Delta}))]$. Its encoder and decoder use feature-wise linear modulation (FiLM) (Perez et al., 2018): a hidden feature h is transformed as $\alpha(c_t) \odot h + b(c_t)$. Given $\widetilde{A}_t = D_{\psi_a}(z_a, c_t)$, its training objective is

$$\mathcal{L}_{\text{action-rep}} = \mathbb{E}_{q_{\phi_a}}[\mathcal{L}_{\text{rec}}(\widetilde{A}_t, A_t)] + \beta_a \mathcal{L}_{\text{KL,free}}. \tag{18}$$

Here $\mathcal{L}_{\text{rec}}$ combines separately masked SmoothL1 losses on position, rotation-vector coordinates, and gripper width with weights $1:1:0.5$ and threshold 0.1. The prior is standard normal; $\mathcal{L}_{\text{KL,free}}$ denotes its KL regularizer with 0.02 nats of free bits per latent dimension. Posterior log-variances are bounded to $[-6, 2]$. Architecture, training budgets, and KL schedules are given in Appendix D.1. The two latents have complementary supervision and conditioning roles; their statistical independence is not assumed.

A.4 Latent Normalization

The world decoder and the conditional action teacher use effect codes in their native coordinates. The policy uses a fixed affine normalization with coordinatewise centering m_e and positive scale r_e:

$$\mathcal{N}_e(z) = \frac{z - m_e}{r_e}, \qquad \mathcal{N}_e^{-1}(u) = m_e + r_e \odot u, \qquad u_e^\star = \mathcal{N}_e(z_e^\star). \tag{19}$$

Division is elementwise. For native-coordinate effect supervision on RoboTwin, $\mathcal{N}_e$ is the identity. Effect codes are restored to native coordinates before world-model decoding.

Action normalization $u = (z - m_a)/r_a$ transforms both posterior means and variances:

$$\begin{aligned} \mu_u &= \frac{\mu_z - m_a}{r_a}, & \sigma_u^2 &= \frac{\sigma_z^2}{r_a^2}, \\ \mu_z &= m_a + r_a \odot \mu_u, & \sigma_z^2 &= r_a^2 \odot \sigma_u^2. \end{aligned} \tag{20}$$

Teacher and policy action distributions are compared in the same normalized coordinates. Trajectory reconstruction restores the predicted action mean to the native coordinates expected by the frozen decoder.

A.5 Effect and Action Prediction

The VLM processes current observations, the instruction, and learnable effect and plan queries Q_e, Q_{plan}:

$$\begin{aligned} (H_{\text{sem}}, H_e, H_{\text{plan}}) &= G_\theta(I_t, \ell; Q_e, Q_{\text{plan}}), \\ \widehat{u}_e &= f_e(H_e), \qquad \widehat{z}_e = \mathcal{N}_e^{-1}(\widehat{u}_e), \\ \widehat{F}_{t+\Delta} &= W_\omega(F_t, \widehat{z}_e). \end{aligned} \tag{21}$$

Thus the feature-level head $g_e = \mathcal{N}_e^{-1} \circ f_e$ restores native effect coordinates, and the complete path satisfies $\pi_e(I_t, \ell) = g_e(H_e)$ with H_e produced by the VLM. The dense action predictor retains spatial transition information by fusing current, future, and difference features at each patch location:

$$\begin{aligned} h_\theta(x, y, d) &= g_\theta([\mathrm{LN}(x); \mathrm{LN}(y); \mathrm{LN}(d)]), \\ T_p &= h_\theta(F_{t,p}, \widehat{F}_{t+\Delta,p}, \widehat{F}_{t+\Delta,p} - F_{t,p}) + b_p^{\text{pos}}. \end{aligned} \tag{22}$$

Here g_θ is a shared MLP and b_p^{pos} is a spatial positional embedding. The visual inputs are detached on this predictor branch, as specified in Appendix D.2. Conditioned plan-query features attend to the transition memory T:

$$\begin{aligned} Q^{(0)} &= \mathrm{Cond}_\theta(H_{\text{plan}}; s_t, \widehat{z}_e, b, \Delta), \\ Q^{(L)} &= \mathrm{Dec}_\theta^{(L)}(Q^{(0)}; K = T, V = T), \qquad L = 3, \\ (\widehat{\mu}_a, \widehat{\sigma}_a^2) &= f_a(\mathrm{Pool}(Q^{(L)})), \\ p_\theta^a &= \mathcal{N}(\widehat{\mu}_a, \mathrm{diag}(\widehat{\sigma}_a^2)). \end{aligned} \tag{23}$$

The embodiment identifier b and horizon Δ specify the trajectory interface. The predicted effect is detached before query conditioning. The head outputs a mean and positive diagonal variances. All inputs to this predictor are available from current information and model predictions at deployment.

A.6 Action-Distribution Supervision

The frozen teacher distribution is $q_a^\star = \mathrm{sg}(q_{\phi_a}(\cdot \mid A_t, c_t))$. For teacher and predicted Gaussians in matching coordinates, forward KL is

$$D_{\mathrm{KL}}(q_a^\star \,\|\, p_\theta^a) = \frac{1}{2}\sum_{j=1}^{d_z}\left[\log\frac{\widehat{\sigma}_{a,j}^2}{\sigma_{a,j}^{\star 2}} + \frac{\sigma_{a,j}^{\star 2} + (\mu_{a,j}^\star - \widehat{\mu}_{a,j})^2}{\widehat{\sigma}_{a,j}^2} - 1\right]. \tag{24}$$

The complete action-representation loss $\mathcal{L}_a$ combines this KL, averaged over the 32 latent dimensions, with mean regression and trajectory reconstruction. Equation equation 27 gives their weights. The reconstruction term passes the predicted action mean through the frozen ActionVAE decoder; gradients reach the action predictor through this decoder despite its fixed parameters. The teacher distribution remains detached throughout policy learning.

A.7 Action Readout and Flow Matching

At inference, the action code is the predicted Gaussian mean, $\widehat{z}_a = \widehat{\mu}_a$. A residual adapter injects it into the noisy-action embeddings:

$$X_a^\tau = \mathrm{Embed}(A^\tau, \tau) + R_\xi(\widehat{z}_a). \tag{25}$$

The adapter output is compatible with the action-token shape and is initialized to zero. The Gaussian variance is trained by posterior matching; inference does not require sampling from this Gaussian. The teacher-to-predicted conditioning curriculum and training dropout are specified in Appendix D.

Let $\mathcal{C}_t = (H_{\mathrm{sem}}, s_t, F_t, \widehat{F}_{t+\Delta})$ denote the expert context. Using a noise-to-data convention, conditional flow matching (Lipman et al., 2023) is written as

$$\begin{aligned} A^\tau &= (1-\tau)\epsilon + \tau A_t, \qquad \epsilon \sim \mathcal{N}(0, I), \quad \tau \in [0,1], \\ \mathcal{L}_{\mathrm{FM}} &= \mathbb{E}\left[\|v_\eta(A^\tau, \tau; \mathcal{C}_t, \widehat{z}_a) - (A_t - \epsilon)\|_F^2\right]. \end{aligned} \tag{26}$$

The velocity network consumes the embeddings in Eq. equation 25. At deployment, integration from $\tau = 0$ to 1 uses ten steps, with the predicted context and action code held fixed. The resulting action chunk is produced by the flow expert; neither posterior encoder nor the ActionVAE decoder is required at inference. Physical action horizons and temporal resampling are described in Appendices B and D.

B Implementation Details

We build on LaWAM (Chen et al., 2026a) with a Qwen3-VL-2B-Instruct backbone (Bai et al., 2025) and a frozen DINOv3 ViT-B/16 encoder (Siméoni et al., 2026). The DINO branch processes 256×256 images into a 16×16 grid of patch features. Both effect and action latents are 32-dimensional. During policy training, the representation teachers remain frozen and their targets are cached offline. We optimize the policy with AdamW, learning-rate warmup, and cosine decay. Policy action chunks span 0.4 s on LIBERO (eight actions at 20 Hz) and 1.2 s on RoboTwin (36 actions at 30 Hz). At inference, the action predictor uses its Gaussian mean, and the flow-matching expert generates each action chunk with ten integration steps.

Policy initialization and data. Each benchmark uses one multitask policy initialized from its released LaWAM SFT checkpoint. LIBERO finetuning uses Original demonstrations for 25,000 additional updates with global batch 256. RoboTwin uses merged demonstrations for 50,000 additional updates with global batch 1,024; the W/P/E allocation is given in Appendix D.

LIBERO rollout initialization. We use the benchmark-provided initial states. The four suites contain 40 evaluation tasks in total, yielding 2,000 rollouts per evaluated policy and evaluation seed under the 50-rollout-per-task protocol. Main policy results are point estimates; variation across independent training runs is not quantified here.

Baseline model version. The GR00T entry in Table 1 refers to N1.6 (NVIDIA GEAR Team, 2025); the original N1 paper (NVIDIA et al., 2025) describes the model family.

Table 6: Per-task success rates (%) on RoboTwin under clean and randomized evaluation. Bold and underlined entries indicate the best and second-best distinct values within each setting, including ties. Averages use all 50 tasks.

Task	Fast-WAM		GigaWorld-Policy		LingBot-VA		$\pi_{0.5}$		Motus		LaWAM		**EAct**	
	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
Adjust Bottle	**100**	**100**	**100**	**100**	98	99	**100**	99	89	93	**100**	**100**	**100**	**100**
Beat Block Hammer	**99**	**100**	86	86	96	98	96	93	95	88	90	93	90	91
Blocks Ranking RGB	**100**	**100**	92	96	98	96	92	85	99	97	97	**100**	97	99
Blocks Ranking Size	94	**98**	44	48	**96**	89	49	26	75	63	93	89	88	93
Click Alarmclock	**100**	**100**	**100**	**100**	**100**	99	98	89	**100**	**100**	**100**	**100**	**100**	**100**
Click Bell	**100**	**100**	**100**	**100**	**100**	**100**	99	66	**100**	**100**	**100**	**100**	**100**	**100**
Dump Bin Bigbin	95	93	92	**100**	87	95	92	97	95	91	**97**	95	90	95
Grab Roller	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Handover Block	99	80	80	80	**100**	**91**	66	57	86	73	96	87	**100**	87
Handover Mic	**100**	**100**	72	72	96	95	98	97	78	63	93	98	95	99
Hanging Mug	**65**	**56**	16	12	21	33	18	17	38	38	51	43	59	30
Lift Pot	**100**	**100**	98	98	**100**	**100**	96	85	96	99	**100**	99	**100**	**100**
Move Can Pot	95	92	76	78	93	93	51	55	34	74	**98**	93	**98**	**96**
Move Pillbottle Pad	**100**	**100**	90	90	**100**	99	84	61	93	96	97	90	99	97
Move Playingcard Away	**100**	**100**	78	72	**100**	**100**	96	84	**100**	96	**100**	**100**	**100**	99
Move Stapler Pad	84	63	92	82	59	71	56	42	83	85	94	**87**	**97**	82
Open Laptop	**100**	**100**	96	98	96	90	90	96	95	91	**100**	**100**	99	**100**
Open Microwave	46	34	74	66	56	80	34	77	**95**	**91**	41	43	66	55
Pick Diverse Bottles	81	88	82	70	93	87	81	71	90	**91**	91	88	**96**	83
Pick Dual Bottles	**100**	94	86	86	**100**	**100**	93	63	96	90	**100**	95	**100**	94
Place A2B Left	94	89	94	88	96	**94**	87	82	88	79	**98**	91	97	87
Place A2B Right	**98**	90	90	92	91	**94**	87	84	91	87	89	**94**	91	**94**
Place Bread Basket	86	**94**	82	82	**94**	**94**	77	64	91	**94**	92	85	91	92
Place Bread Skillet	92	**94**	94	90	**95**	92	85	66	86	83	90	83	94	90
Place Burger Fries	**98**	92	**98**	96	95	**98**	94	87	**98**	**98**	93	96	95	97
Place Can Basket	72	67	78	74	83	**80**	62	62	81	76	**92**	65	79	65
Place Cans Plasticbox	98	**100**	**100**	**100**	**100**	98	94	84	98	94	**100**	95	**100**	95
Place Container Plate	98	**100**	98	96	97	98	99	95	98	99	**100**	**100**	96	98
Place Dual Shoes	88	88	96	84	94	81	75	75	93	87	**98**	94	96	**97**
Place Empty Cup	**100**	**100**	90	90	**100**	**100**	**100**	99	99	98	99	**100**	99	**100**
Place Fan	**100**	**96**	92	94	95	88	87	85	91	87	92	93	98	95
Place Mouse Pad	84	89	88	90	**92**	**93**	60	39	66	68	91	84	91	84
Place Object Basket	82	90	90	**92**	89	85	80	76	81	87	**92**	90	89	88
Place Object Scale	83	88	88	80	**98**	87	86	80	88	85	95	88	**98**	**91**
Place Object Stand	88	91	**100**	**98**	99	91	91	85	98	97	92	93	97	92
Place Phone Stand	**98**	**98**	82	72	**98**	97	81	81	87	86	93	94	95	97
Place Shoe	97	98	98	96	99	97	92	93	99	97	**100**	**100**	**100**	**100**
Press Stapler	**98**	**100**	96	96	82	88	87	83	93	98	**98**	97	97	**100**
Put Bottles Dustbin	93	82	72	70	82	87	84	79	81	79	**94**	**92**	**94**	89
Put Object Cabinet	**94**	**82**	74	74	87	80	80	79	88	71	90	**82**	88	81
Rotate QRcode	**95**	94	90	84	92	**95**	89	87	89	73	94	89	94	**95**
Scan Object	**96**	86	60	64	90	89	72	65	67	66	**96**	**90**	91	**90**
Shake Bottle	**100**	**100**	**100**	**100**	**100**	98	99	97	**100**	97	**100**	**100**	**100**	**100**
Shake Bottle Horizontally	**100**	**100**	**100**	98	99	99	99	99	**100**	98	**100**	**100**	**100**	**100**
Stack Blocks Three	98	96	70	78	**100**	**99**	91	76	91	95	90	75	89	79
Stack Blocks Two	**100**	**100**	**100**	94	**100**	99	97	**100**	**100**	98	**100**	97	**100**	99
Stack Bowls Three	77	86	70	72	89	79	77	71	79	**87**	**90**	80	80	77
Stack Bowls Two	90	96	96	92	98	**99**	95	96	98	98	**100**	**99**	97	92
Stamp Seal	78	86	96	**98**	**98**	95	79	55	93	92	89	88	89	87
Turn Switch	66	56	82	**84**	54	57	62	54	**84**	78	47	56	53	53
Average	91.98	90.52	86.36	85.04	91.50	**90.92**	82.74	76.76	88.66	87.02	92.64	89.80	**93.04**	90.08

C Detailed RoboTwin Results

Table 6 reports results for all 50 tasks. Baseline values follow LaWAM (Chen et al., 2026a).

D EAct Training Pipeline

We first prepare frozen representation teachers, then apply world alignment (W), plan learning (P), and joint optimization (E) on both LIBERO and RoboTwin. Table 7 reports per-phase optimizer updates after gradient accumulation, excluding initialization checkpoints and teacher training.

Platform	Phase	Updates	Warmup	Batch	VLM LR	Other LR
LIBERO	W	2,000	120	256	Frozen	10^{-4}
	P	8,000	480	256	10^{-5}	10^{-4}
	E	15,000	900	256	10^{-5}	10^{-4}
RoboTwin	W	8,000	480	1,024	Frozen	10^{-4}
	P	12,000	720	1,024	5×10^{-5}	10^{-4}
	E	30,000	1,800	1,024	10^{-4}	10^{-4}

Table 7: **Platform-specific policy finetuning schedules.** Batch denotes global batch size. Learning rates are peak values and apply only to trainable parameters; "Other LR" covers the active prediction heads, world decoder, residual adapter, and action expert. Phase freezing is described below.

D.1 Representation Preparation

Visual-transition teacher. For LIBERO, we initialize from the released LaWAM latent model and learn the geometry-regularized effect representation for 40 epochs on 1.2-second windows. DINOv3 remains frozen. The effective batch size is 512 (2 GPUs $\times 64$ samples $\times 4$ accumulation steps). AdamW uses learning rates of 3×10^{-5} for the inverse-dynamics, variational, and state-prediction modules and 10^{-5} for the world decoder, with weight decay 10^{-2}. A one-epoch learning-rate warmup precedes cosine decay to 10% of the peak rates. The Gaussian-prior weight is 10^{-5}. The neighborhood weight is zero for the first epoch and increases linearly to 0.05 over the next four epochs. We select the epoch-39 effect checkpoint.

For RoboTwin, we use the same base architecture and loss formulation. We train for 40 epochs with BF16 mixed precision and an effective batch size of 512 (two GPUs, 64 samples per GPU, and four accumulation steps). AdamW uses learning rates of 3×10^{-5} for the effect encoder, variational head, and state decoder, and 10^{-5} for the world decoder, with $(\beta_1, \beta_2) = (0.9, 0.95)$ and weight decay 10^{-2}. The prior-KL weight is 10^{-5}. The neighborhood weight is zero for the first epoch and increases linearly to 0.05 over the next four epochs. A one-epoch learning-rate warmup precedes cosine decay to 10% of the peak rates. We select the epoch-9 checkpoint using validation reconstruction and geometry criteria without consulting the test split.

Trajectory teacher. The conditional TCN uses hidden and condition widths of 128, kernel size 3, and dilations $(1, 2, 4, 8)$. We initialize the ActionVAE from a deterministic conditional autoencoder trained with learning rate 3×10^{-4} and weight decay 10^{-4}, with budgets of 100 epochs on LIBERO and 50,000 updates on RoboTwin. The subsequent variational training budget is 10,000 updates, using AdamW at 10^{-4} with weight decay 10^{-4}. The latent prior is standard normal. LIBERO ramps the KL weight to 0.2 over 200 updates without a mean-only warmup; RoboTwin uses 500 mean-only updates followed by a 2,500-update ramp to 10^{-3}. Both use 0.02 nats of free bits per latent dimension and log-variance bounds $[-6, 2]$. Reconstruction uses separately masked SmoothL1 losses (threshold 0.1) for position, rotation-vector coordinates, and gripper width, weighted $1 : 1 : 0.5$. The selected variational teachers are LIBERO step 9,889 and RoboTwin step 5,000; these selections are distinct from their training budgets.

Offline target construction. After freezing the teachers, we cache the visual-condition mean and the action posterior mean and log-variance for each training window. A fixed affine normalization maps latent targets into policy coordinates; action variances are transformed along with their means. Codes are restored to native coordinates before entering the world or trajectory decoder. The frozen ActionVAE decoder remains available during policy training for trajectory supervision, but is not the deployed action generator.

D.2 Policy Objectives and Phase Roles

Let $\mathcal{L}_e$ denote supervision of the effect prediction: SmoothL1 with threshold 0.1 on normalized effect means for LIBERO, or MSE on native LaWAM latent means for RoboTwin. The future-feature loss

$\mathcal{L}_f$ is MSE. The action-representation objective in Eq. equation 8 is

$$\begin{aligned}\mathcal{L}_a &= \frac{1}{32}\mathbb{E}[D_{\mathrm{KL}}(q_a^\star \,\|\, p_\theta^a)] + 0.5\mathcal{L}_\mu + 0.2\mathcal{L}_{\mathrm{traj}},\\ \mathcal{L}_\mu &= \mathrm{SmoothL1}_{0.1}(\widehat{\mu}_a, \mu_a^\star).\end{aligned} \tag{27}$$

Here both Gaussians use the same normalized coordinates, and the KL is averaged over the batch and latent dimensions. $\mathcal{L}_{\mathrm{traj}}$ reconstructs the demonstrated trajectory through the frozen ActionVAE decoder using the predicted action mean, current state, and detached predicted visual condition. Its position, rotation, and gripper terms use the same $1:1:0.5$ weighting. Although the decoder weights are frozen, gradients propagate through it to the predicted action latent.

Phase	$\mathcal{L}_{\mathrm{FM}}$	$\mathcal{L}_e$	$\mathcal{L}_a$	$\mathcal{L}_f$
W	0	1	0	1
P	1	0.1	0.5	0.1
E	1	0.1	0.25	0.1

Table 8: **Weights in the policy training objective.** Each row defines the weighted sum of the four losses. The definition of $\mathcal{L}_e$ follows the platform's effect teacher. The action-plan weights multiply the complete objective in Eq. equation 27.

In W, only the effect prediction head and world decoder are updated; the VLM, action predictor, and action expert are frozen. In P, the VLM, visual-condition predictor, action predictor, and residual adapter are trainable, while the world decoder and action-expert weights remain frozen. The frozen expert can still transmit flow-loss gradients to its trainable inputs. E additionally updates the world decoder and action expert. DINOv3, posterior teachers, and the trajectory decoder remain frozen in all policy phases. The action predictor receives detached visual features and effect codes; its losses therefore do not backpropagate through these inputs to the visual prediction path. The effect and future-feature objectives directly supervise that path.

D.3 LIBERO Finetuning

Starting from the released pretrained LaWAM model, we first run 25,000 updates under the baseline SFT schedule. We then train EAct for 2,000 W updates, 8,000 P updates, and 15,000 E updates, totaling 25,000 additional policy updates. P and E start from the preceding phase's monitoring-loss-selected checkpoint, each with a fresh optimizer and learning-rate schedule. Four GPUs, per-device batch 16, and four accumulation steps give global batch 256. The policy retains the first 16 language blocks; the VLM is frozen in W and its configured trainable modules are updated in P and E.

During P, action-latent residual conditioning is disabled for the first 6,000 updates. Over updates 6,000–6,999, the probability of conditioning on a teacher action code decreases linearly from 0.5 to zero, and the predicted residual input is detached from the action predictor. From update 7,000 onward, predicted codes drive the residual and receive flow-loss gradients. E uses predicted codes throughout, with 0.1 residual-conditioning dropout during training. Eight physical control intervals at 20 Hz define each 0.4-second policy chunk; the frozen ActionVAE interface resamples these into 24 trajectory slots without changing the physical duration.

D.4 RoboTwin Finetuning

Starting from the released pretrained LaWAM model, we train EAct for 8,000 W updates, 12,000 P updates and 30,000 E updates, totaling 50,000 additional policy updates. The visual-condition mapping, dense action predictor, and zero-initialized residual are optimized in P, followed by joint policy optimization in E. Eight H200 GPUs, per-device batch 32, and four accumulation steps give global batch 1,024. Training uses dense stride-one windows from the merged RoboTwin data, with 36 physical actions at 30 Hz per 1.2-second chunk.

The residual curriculum uses global Stage 2 update counts, including W. Conditioning is disabled throughout W and the first 4,000 P updates. The teacher-conditioning probability decreases from 0.5 to zero over global updates 12,000–13,999 (P-local updates 4,000–5,999), and flow gradients enter the action predictor from global update 14,000 (P-local update 6,000). E uses predicted action

codes with 0.1 conditioning dropout. The VLM is frozen in W, with peak learning rate 5×10^{-5} in P and 10^{-4} in E. The world decoder is trained in W, frozen in P, and unfrozen in E; the action expert remains frozen through W/P and is unfrozen in E.

D.5 Optimization and Checkpoint Selection

All policies use BF16 mixed precision, AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$ and $\epsilon = 10^{-8}$, and gradient-norm clipping at 1.0. LIBERO and RoboTwin use weight decay 10^{-8} and phase-wise cosine schedules following the warmups in Table 7. Their learning-rate multiplier decays to 0.005 of each group's peak rate, corresponding to 5×10^{-7} for a group peaking at 10^{-4}.

Simulation phases evaluate the total training objective every 500 updates over 20 monitoring batches and select the lowest-loss checkpoint for the next phase. LIBERO follows the inherited all-row training convention: its tail monitoring subset is not an independent held-out generalization set. RoboTwin uses episode-disjoint training and validation splits.

E Additional Experiments and Analysis Details

E.1 Effect Neighborhood Evaluation

We evaluate 2,975 non-overlapping transitions from 650 held-out LIBERO episodes across 130 tasks. This representation-analysis pool is distinct from the 40-task policy evaluation described in Section 4.1. The full effect encoder is compared with a matched-budget counterpart trained without the neighborhood loss $\mathcal{L}_{\text{nbr}}$. Let $\mathcal{N}_k^v(i)$ and $\mathcal{N}_k^e(i)$ denote the k nearest neighbors of transition i, excluding i itself, under frozen DINOv3 visual-transition descriptors and effect posterior means, respectively. Neighborhood preservation is measured by

$$\text{Overlap@}k(i) = \frac{|\mathcal{N}_k^v(i) \cap \mathcal{N}_k^e(i)|}{k}. \tag{28}$$

We average scores within episodes, then within tasks, and finally across tasks with equal task weights.

Table 3 reports within-task, global, and context-matched neighborhoods. The context-matched evaluation restricts candidates to the 64 transitions with the most similar initial DINOv3 visual contexts. We also report within-task local Spearman distance-rank correlation, ρ_{local}. Figure 4 shows the per-task within-task Overlap@5 comparison: regularization improves 100 of 130 tasks, with a mean gain of 2.59 percentage points.

E.2 Execution-Residual Probes

Targets. For eight future waypoints over 0.4 seconds, let $(\mathbf{p}_i, R_i, g_i)$ denote position, orientation, and gripper width, with poses expressed in the initial end-effector frame, so that $\mathbf{p}_0 = 0$ and $R_0 = I$. We remove the endpoint-interpolated motion:

$$\begin{aligned} \mathbf{r}_i^p &= \mathbf{p}_i - \tfrac{i}{8}\mathbf{p}_8, \\ \bar{R}_i &= \text{Exp}\big(\tfrac{i}{8}\,\text{Log}(R_8)\big)\,, \\ \mathbf{r}_i^R &= \text{Log}(\bar{R}_i^\top R_i), \qquad i = 1, \dots, 7. \end{aligned} \tag{29}$$

Here, $\text{Exp} : \mathbb{R}^3 \to \text{SO}(3)$ maps rotation vectors to rotation matrices, and Log maps rotations back to rotation vectors. Together with $\Delta g_i = g_i - g_0$, the target is $\mathbf{r} = [\mathbf{r}_{1:7}^p; \mathbf{r}_{1:7}^R; \Delta g_{1:8}] \in \mathbb{R}^{50}$.

Inputs and fitting. We predict $\mathbf{r}$ from $[\mathbf{s}; \mathbf{z}] \in \mathbb{R}^{46}$, combining the 14-dimensional current state with a 32-dimensional latent. The effect and action teacher inputs are posterior means $z_e^\star$ and $z_a^\star$; their predicted counterparts are $\widehat{z}_e$ and $\widehat{z}_a$. The LaWAM baseline uses the corresponding posterior mean and policy prediction. All representations are frozen, and all state-plus-latent inputs have the same dimensionality. We fit standardized Ridge regressors and MSE-trained MLPs, with a state-only probe as a baseline.

Evaluation. For this in-distribution analysis, 1,941 windows from 16 Stage-2 training episodes across eight tasks are held out from probe fitting. Table 4 reports execution-residual task-macro R^2; MLP results are the mean and sample standard deviation over three probe seeds. The predicted action latent improves Ridge R^2 over LaWAM by 0.1648, with gains across all eight tasks. Because these episodes are drawn from the Stage-2 training distribution, the probe results measure encoded execution information rather than generalization to unseen policy tasks.

E.3 Component-Ablation Protocols and Suite Results

The interventions in Section 4.3 test the following design choices. Table 9 gives the policy success rates underlying Figure 3.

- **Without the action token:** remove the action-latent prediction and conditioning branch, retaining effect-guided control.
- **Predictor without future input:** at test time, replace the predicted future features with current features only in the action predictor. The action expert continues to receive predicted future features. This is an input intervention on a trained model, testing the predictor's use of anticipated transitions.
- **Without token injection:** retain action-latent supervision but disable its residual conditioning of the expert.

Variant	Long	Goal	Object	Spatial	Average
Full EAct	97.6	98.4	100.0	99.8	98.95
w/o action token	95.4	98.2	99.2	99.8	98.15
Predictor w/o future	94.6	97.4	99.4	98.6	97.50
w/o token injection	95.6	95.8	99.2	98.6	97.30

Table 9: **LIBERO policy ablations by suite.** Values are success rates (%). Each ablation uses 500 rollouts per suite. Averages use the unrounded suite scores; the full-model average of 98.95% is reported as 99.0% in the main performance table.

E.4 Additional Predictor and Teacher Comparisons

Predictor architecture. To examine the choice of action-prediction architecture, we replace the cross-attention predictor with an MLP. This variant achieves 98.00% average success, compared with 98.95% for full EAct (Table 10).

Action-teacher transfer. We reuse a frozen RoboTwin-trained ActionVAE while training the downstream policy on LIBERO. Fixed embodiment and temporal adapters connect the source teacher to LIBERO, retaining its source-compatible LaWAM conditioning. Without finetuning the ActionVAE on LIBERO, the resulting policy achieves 97.55% average success on the four original LIBERO suites. This evaluates reuse of the trajectory teacher with downstream policy adaptation, rather than zero-shot deployment of a RoboTwin policy on LIBERO.

Variant	Long	Goal	Object	Spatial	Average
Full EAct	97.6	98.4	100.0	99.8	98.95
MLP predictor	96.0	96.6	99.8	99.6	98.00
RoboTwin ActionVAE	95.4	96.4	99.6	98.8	97.55

Table 10: **Additional LIBERO comparisons.** Values are suite success rates (%) and their average. The MLP variant changes the predictor architecture; the RoboTwin ActionVAE variant transfers the action teacher and trains the downstream policy on LIBERO.

E.5 LEARNING-EFFICIENCY PROTOCOL

The learning-efficiency study evaluates the effect-only variant after 2,500, 5,000, and 12,500 updates of a 25,000-update Stage-2 schedule. The 80% reduction compares 2,500 effect-only updates with 12,500 original-latent baseline updates: $1 - 2{,}500/12{,}500 = 0.8$. Representation training, target caching, and initialization costs are excluded, so this comparison does not measure total training compute.