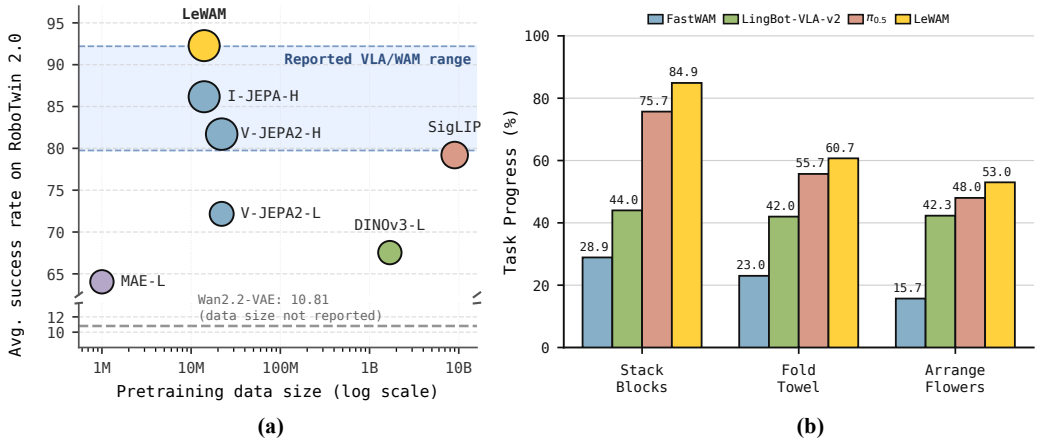
LATENT EVOLVING WORLD ACTION MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) jointly model action generation and environment dynamics and are mostly built on pretrained Video Diffusion Models (VDMs). In VDM-based WAMs, observations are first encoded by a VAE, and the resulting compressed latents are then processed by large video diffusion backbones to extract effective features for action generation. However, this paradigm ties WAM performance and training cost to large-scale video generation pretraining, limiting WAM efficiency and scalability. In this paper, we theoretically and empirically investigate how visual representations affect action generation in WAMs. Our results show that predictive embeddings from Joint-Embedding Predictive Architecture (JEPA) encoders better support action generation than compressed VAE latents, with I-JEPA performing best in our encoder comparison. Based on these findings, we propose LeWAM, which conditions action generation on JEPA embeddings and models environment evolution by predicting future embeddings in the same space, without relying on a video diffusion backbone. We further find that imitation learning matches demonstrated actions but does not distinguish better actions from worse ones, even though small action deviations can greatly affect task success. To address this limitation without additional environment interaction or the human oversight required for resets and safety, we introduce Demonstration-Guided DPO (DemoDPO), an offline preference refinement stage that derives preference supervision directly from demonstrations. With only 0.4B trainable parameters, LeWAM achieves an average success rate of 92.28% on RoboTwin 2.0, comparable to that of state-of-the-art VLAs and WAMs, and maintains practical effectiveness on real-world manipulation tasks.



(a) **(b)**

Figure 1: *(a)* **Frozen vision encoder comparison and LeWAM performance on RoboTwin 2.0.** Frozen encoders are evaluated with the same simple action predictor architecture and training recipe. I-JEPA-H, pretrained only on ImageNet-22K, performs best among the evaluated encoders, while LeWAM achieves an average success rate of 92.28%. *(b)* **Real-world task progress across three manipulation tasks.** LeWAM achieves the highest task progress on all three tasks, outperforming $\pi_{0.5}$, LingBot-VLA-v2, and FastWAM.

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.

1 INTRODUCTION

Vision-Language-Action (VLA) models (Brohan et al., 2022; Belkhale et al., 2024; Kim et al., 2024; Wen et al., 2025; Fan et al., 2025; Physical Intelligence et al., 2025; Chen et al., 2025a; Black et al., 2026; GigaBrain Team et al., 2026; Zhou et al., 2025) have become a widely used paradigm for generalist robotic policies, mapping visual observations and language instructions directly to actions via behavioral cloning. By leveraging large pretrained vision-language backbones, VLAs generalize across language instructions, but they do not explicitly model how the physical world evolves under actions. World Action Models (WAMs) (Ye et al., 2026b;a; Ma et al., 2026; Yuan et al., 2026) address this limitation by coupling action prediction with visual dynamics learning, often using Video Diffusion Model (VDM) backbones to model how actions change the physical scene. These WAMs provide additional supervision for embodied policies and report competitive results on manipulation tasks.

Despite these advances, recent WAMs reveal a tension between how they are trained and deployed. Some WAMs retain explicit joint video and action generation at inference time (Ye et al., 2026b; Bi et al., 2026; Ma et al., 2026), but action-centered WAMs increasingly remove future visual generation during deployment (Ye et al., 2026a; Yuan et al., 2026). These WAMs still benefit from video supervision during training, but at test time often generate actions directly from the current observation and task instruction. At deployment, they use VLA-style inference, predicting actions directly without forecasting future state changes. For VDM-based WAMs, the visual representation can therefore become a bottleneck for action generation: compressed VAE posterior means can be poorly suited to lightweight action predictors, and extracting useful features from them relies on a large pretrained video diffusion backbone, limiting efficiency and scalability.

Prior studies provide evidence on visual representations for action generation, but their settings are not directly comparable. Existing encoder comparisons cover a limited set of representations or evaluate them as additions to VLA features (Wu et al., 2024; Egbe et al., 2025; Hu et al., 2024; Miao et al., 2026; Wang et al., 2025). Existing latent world models use predictive embeddings for goal-conditioned planning rather than direct action generation (Assran et al., 2025; Maes et al., 2026). Existing VLAs and VDM-based WAMs rely on different visual pipelines, using pretrained VLM features and VAE latents processed by video diffusion backbones, respectively (Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025; Wen et al., 2025; Zheng et al., 2025; Ye et al., 2026b; Bi et al., 2026; Ma et al., 2026; Yuan et al., 2026; Ye et al., 2026a). None of these studies compares representative frozen encoders learned through different pretraining objectives as direct inputs to the same action predictor. These objectives include reconstruction (He et al., 2022; Chen et al., 2024), self-supervised feature learning (Caron et al., 2021; Zhang et al., 2019; Qi et al., 2019), vision-language alignment (Zhai et al., 2023), and image or video prediction (Assran et al., 2023; 2025; Zhang et al., 2023). Such a comparison is necessary to isolate the contribution of the visual representation from the policy and training procedure.

We therefore analyze WAMs built on VDMs from the perspective of information theory and compare representative frozen encoders under the same training recipe. Our analysis separates the action information a representation retains from the part a downstream model can use, while the controlled comparison shows that compressed VAE latents are poorly suited to direct action generation and that I-JEPA performs best among the evaluated encoders. These findings motivate LeWAM, a **L**atent **e**volving **W**orld **A**ction **M**odel that performs both action generation and world modeling in JEPA embedding space. To jointly learn action generation and future embedding prediction conditioned on actions, LeWAM organizes the current embedding, noisy and demonstrated action tokens, and future embedding queries in a unified token sequence and controls the information flow with a structured attention mask. This design allows both objectives to be learned by a single predictor without information leakage. To further exploit the information distributed across the frozen encoder, we introduce AdaFuse, which adaptively combines representations from different encoder layers for each feature dimension. At inference time, LeWAM removes the demonstrated actions and future queries and generates actions only from the current observation and task context.

We observe that LeWAM generates actions close to demonstrations on the validation set, yet small deviations during interaction with the environment can cause task failure. We further find that allowing the policy multiple attempts increases its success rate. This suggests that successful action trajectories lie within the policy distribution but receive insufficient probability mass. We therefore aim

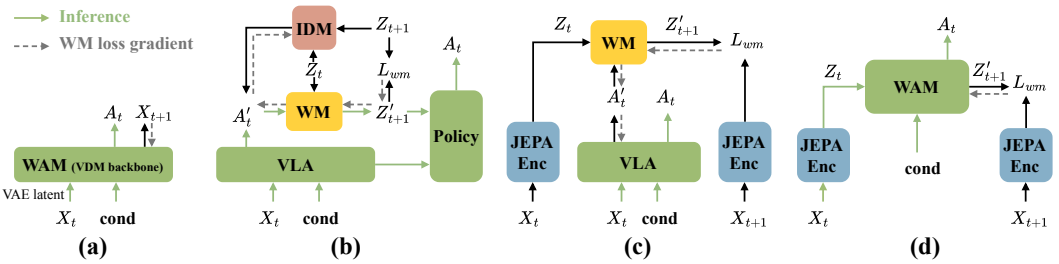


Figure 2: Comparison of WAM architectures. **(a) Action-centered VDM-based WAMs** use compressed VAE latents as visual inputs and jointly learn action generation and future frame prediction, but generate only actions at inference. **(b) LaWAM** (Chen et al., 2026) predicts latent actions from VLM representations and uses a separate world model to predict future DINOv3 embeddings, which condition action generation at inference. **(c) VLA-JEPA** (Sun et al., 2026) generates actions from VLM representations and uses latent action tokens to condition a separate world model supervised by V-JEPA2 embeddings during training. **(d) LeWAM** ses frozen I-JEPA embeddings as both visual inputs for action generation and targets for action-conditioned future prediction, learning both tasks with a single predictor while generating only actions at inference.

to shift the policy distribution toward these trajectories. A natural approach is to maximize expected return through online reinforcement learning with rollout rewards (Ren et al., 2025; Yi et al., 2026). However, on physical robots, this requires repeated environment interaction and human oversight for resets and safety, making it difficult to scale (Eysenbach et al., 2017; Sharma et al., 2021; Walke et al., 2023; Patil et al., 2024). To avoid this requirement, we introduce Demonstration-Guided DPO (DemoDPO) as the offline preference refinement stage of LeWAM. DemoDPO uses demonstrations to construct preferences among action candidates from a frozen reference policy, providing offline preference supervision without additional environment interaction or manually designed rewards.

Our contributions are summarized as follows:

- We theoretically and empirically study visual representations for action generation in WAMs, identify I-JEPA as an effective visual encoder for action generation, and further examine its representations through attention visualization and probing.
- We propose LeWAM, which combines AdaFuse with joint action generation and future embedding prediction in a lightweight model, while generating only actions at inference.
- We refine LeWAM with DemoDPO, using demonstrations to provide preference supervision without additional environment interaction or manually designed rewards.

2 Related Work

Visual Representations and World Modeling for Action Generation. Diffusion Policy (Chi et al., 2025) and Octo (Octo Model Team et al., 2024) generate action sequences through conditional denoising without an explicit world model. VDM-based WAMs typically couple action learning with future video modeling through pretrained video diffusion backbones, as illustrated in Figure 2*(a)* (Ye et al., 2026b; Bi et al., 2026; Ma et al., 2026; Yuan et al., 2026; Ye et al., 2026a). Recent methods instead predict future states in pretrained representation spaces. VLA-JEPA uses latent action tokens produced by a VLM to condition a separate world model trained against frozen V-JEPA2 targets (Sun et al., 2026). The flow matching action head is conditioned on VLM representations, whereas V-JEPA2 embeddings supervise only dynamics learning and do not serve as direct visual inputs for action generation, as shown in Figure 2*(c)*. LaWAM first learns a latent action model in a frozen DINOv3 space and then factorizes control into a VLA that predicts a latent action, a separate latent world model that decodes it into a future visual subgoal, and an action expert conditioned on that subgoal (Chen et al., 2026). Future prediction therefore remains an explicit intermediate at deployment, as shown in Figure 2*(b)*. LeWAM instead uses frozen I-JEPA embeddings as both the direct visual input for action generation and the target for future prediction. A single predictor jointly generates actions and predicts future embeddings conditioned on demonstrated action prefixes during training, while future prediction is removed at deployment, as shown in Figure 2*(d)*.

Robot Policy Refinement beyond Imitation. Online reinforcement learning has been used to refine diffusion and flow matching policies using rewards collected through environment interaction (Ren et al., 2025; Yi et al., 2026). However, applying these methods to physical robots requires repeated environment rollouts, reliable task rewards, and human oversight for resets and safety, making them difficult to scale across tasks (Eysenbach et al., 2017; Sharma et al., 2021; Walke et al., 2023; Patil et al., 2024). Preference-based methods provide an alternative, but existing approaches derive supervision from predefined task rewards or reward-based evaluators (Zhang et al., 2024a; Shan et al., 2025), model-generated evaluation criteria (Zhang et al., 2024b), or direct human feedback, including pairwise comparisons, interventions, and manually specified behavior preferences (Chen et al., 2025b; Xia et al., 2026; Vatsa et al., 2026). These signals still require additional supervision or interaction beyond the original offline demonstrations. DemoDPO instead uses each existing demonstration to rank a group of candidates sampled from a frozen reference policy and filters groups without sufficiently distinct preferences. It therefore enables offline preference refinement without additional environment interaction, reward supervision, or human preference annotation.

3 Representation-Guided Latent World Action Modeling

In this section, we analyze visual representations for action generation and the limitations of VDM-based WAMs. We then present LeWAM and its DemoDPO preference refinement stage.

3.1 Problem Formulation

We study visuomotor policy learning from expert demonstrations. At time t, the policy maps a visual observation X_t and task context c_t to an expert action chunk $A_t = (a_t, \dots, a_{t+H_\mathrm{a}-1})$, where H_a is the number of control steps per generation pass. LeWAM parameterizes the conditional policy $\pi_\theta(A_t \mid X_t, c_t)$ using flow matching Lipman et al. (2023). World Action Models augment this objective by modeling environment evolution over the same horizon. We select N future offsets $0 < \delta_1 < \cdots < \delta_N \le H_\mathrm{a}$ and define $Z_i = E(X_{t+\delta_i})$, where the visual representation function E produces $Z_i \in \mathbb{R}^{hw \times d}$ as an $h \times w$ grid of d-dimensional patch embeddings. LeWAM jointly generates A_t and predicts $\{Z_i\}_{i=1}^N$, as detailed in Section 3.3. Below, we omit time indices and let X, A, c, and $Z = E(X)$ denote random variables drawn from the demonstrations. Unless stated otherwise, all probabilities, expectations, and information quantities refer to their joint distribution.

3.2 Action-Sufficient World Modeling

We analyze how a visual representation affects action uncertainty.

Assumption 1 (Action sufficiency)**.** *The factors $S = S(X)$ form a sufficient statistic for the action given the context, that is, $p(A \mid X, c) = p(A \mid S, c)$.*

Proposition 1 (Action relevance gap)**.** *Under Assumption 1, let $Z = E(X)$, where E is deterministic. When the conditional action entropies are finite, the additional action uncertainty satisfies*

$$\Delta_E(A) := H(A \mid Z, c) - H(A \mid X, c) = I(A; X \mid Z, c) = I(A; S \mid Z, c) \ge 0, \tag{1}$$

where H denotes entropy for discrete actions and differential entropy for continuous actions, and I denotes conditional mutual information.

A useful representation for action generation should retain the information in S relevant to A.

We next examine VAE compression and the reliance on large video diffusion backbones in VDM-based WAMs. First, a linear Gaussian analysis makes explicit how VAE training can suppress observation directions in the posterior mean (Wang & Ziyin, 2022).

Proposition 2 (VAE spectral compression)**.** *For fixed c, consider the linear Gaussian VAE in Appendix A.2, optimized to a global minimum with latent width k, KL weight $\beta_\mathrm{vae} > 0$, and fixed decoder variance $\sigma^2_\mathrm{dec} > 0$. Let (λ_i, u_i) denote the observation covariance eigenpairs in decreasing eigenvalue order, and set $\lambda_\mathrm{cut} = \beta_\mathrm{vae}\sigma^2_\mathrm{dec}$. The posterior mean Z_vae has effective dimension*

$$r_\mathrm{eff} := \operatorname{rank} \operatorname{Cov}(Z_\mathrm{vae} \mid c) = \min\big(k, \#\{i : \lambda_i > \lambda_\mathrm{cut}\}\big). \tag{2}$$

Let $J = \{i \le k : \lambda_i > \lambda_\mathrm{cut}\}$. If $A = B_c X + \xi$, where ξ is independent Gaussian noise with positive definite covariance, then $\Delta_E(A) > 0$ exactly when $B_c u_i \neq 0$ for some $i \notin J$.

Proposition 2 and Corollary 1 show that reconstruction based selection need not retain action-relevant directions. Appendix A.3 further distinguishes information loss from the difficulty of extracting retained information with a fixed predictor (Xu et al., 2020). This distinction motivates our controlled comparison of frozen representations for action generation in Section 4.2.

Second, VDM-based WAMs rely on a large video diffusion backbone pretrained at scale for useful visual features. Indeed, Wang et al. (2026) show that such a backbone can be repurposed into a strong feedforward perception model. This result demonstrates the value of video generation pretraining, but does not establish the effectiveness of raw VAE latents as direct representations for action generation. The diffusion backbone extracts the information retained in Z_{vae} and exploits learned visual priors. Section 4.2 evaluates whether a lightweight predictor on a frozen JEPA encoder matches these systems without such a backbone.

3.3 World Action Modeling in Representation Space

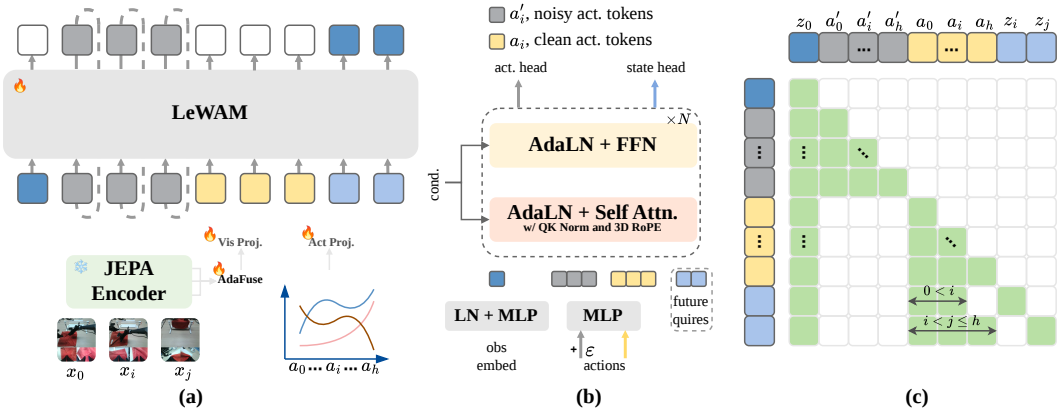


Figure 3: LeWAM architecture for joint action generation and future embedding prediction. *(a)* A frozen JEPA encoder extracts multilayer embeddings, and AdaFuse adaptively combines them. *(b)* The predictor combines observation and action tokens with future queries to predict action velocities and future embeddings. *(c)* The mask prevents clean actions from leaking into action prediction and restricts each future query to the action prefix for its randomly sampled horizon.

Figure 3 shows how LeWAM learns an action policy and a representation space world model within a single predictor:

$$\pi_\theta(A_t \mid X_t, c_t), \qquad p_\theta\left(Z_i \mid X_t, A_t^{(\delta_i)}, c_t\right), \quad i = 1, \ldots, N, \tag{3}$$

where $A_t^{(\delta_i)} = (a_t, \ldots, a_{t+\delta_i-1})$ is the clean action prefix leading to $X_{t+\delta_i}$ and $Z_i = E(X_{t+\delta_i})$. The first conditional generates an action chunk from the current observation, while the second predicts how the visual embedding evolves under the demonstrated action prefix. Given a training segment starting at time t, let $E_\ell(X) \in \mathbb{R}^{hw \times d}$ denote the patch embeddings from layer ℓ of a frozen JEPA encoder with L layers. AdaFuse learns logits $\Gamma \in \mathbb{R}^{L \times d}$, normalizes them across layers separately for each feature dimension, and computes

$$F_{\text{Ada}}(X) = \text{LN}_{\text{out}}\left(\sum_{\ell=1}^{L} w_\ell \odot \text{LN}_{\text{in}}(E_\ell(X))\right), \qquad w_{\ell,k} = \frac{\exp(\Gamma_{\ell,k})}{\sum_{r=1}^{L} \exp(\Gamma_{r,k})}, \tag{4}$$

where $\odot$ denotes elementwise multiplication, and LN_{in} and LN_{out} apply LayerNorm before and after fusion. The channelwise fusion weights are shared across patch tokens and observations. We use AdaFuse for the current observation and obtain future targets directly from the frozen encoder: $Z_0 = F_{\text{Ada}}(X_t)$ and $Z_i = E(X_{t+\delta_i})$, where E denotes the frozen encoder. For conditional flow matching, we sample $\tau \sim \text{Unif}[0, 1]$ and form

$$A'_\tau = (1 - \tau)A_t + \tau\epsilon, \qquad \epsilon \sim \mathcal{N}(0, \mathbf{I}), \qquad v_\tau = \epsilon - A_t. \tag{5}$$

To jointly model action generation and future embedding prediction, LeWAM processes the token sequence $[Z_0, A'_\tau, A_t, Q_{1:N}]$ with a single predictor. Each Q_i is formed by repeating a learned query token over a patch grid matching Z_0 and adding learned position embeddings. Each attention layer uses QK normalization and 3D RoPE with one temporal and two spatial coordinates. Z_0 and Q_i use

their spatiotemporal positions, whereas noisy and clean action tokens use their temporal positions with both spatial coordinates set to -1. A structured attention mask allows each noisy action token to attend only to Z_0 and its causal noisy action prefix, makes the clean action tokens causal, and allows Q_i to attend only to Z_0, its own query grid, and the first δ_i clean actions. This prevents clean actions from leaking into action generation and actions after δ_i from affecting the prediction of Z_i, either directly or through earlier clean action states. The causal noisy action stream also supports different action horizons at inference. Under this mask, the noisy action stream realizes $\pi_\theta(A_t \mid X_t, c_t)$, while the future query stream realizes $p_\theta(Z_i \mid X_t, A_t^{(\delta_i)}, c_t)$.

Separate heads map the hidden states of the noisy action tokens to the flow velocity $\hat{v}_\theta$ and those of the future query tokens to the predicted future embeddings $\hat{Z}_{1:N}$. We optimize both objectives with

$$\mathcal{L}_{\text{act}} = \mathbb{E}\left[\|\hat{v}_\theta - v_\tau\|_2^2\right], \qquad \mathcal{L}_{\text{wm}} = \mathbb{E}_i\left[\left\|\hat{Z}_i - \text{sg}(Z_i)\right\|_2^2\right], \tag{6}$$

where $\mathbb{E}_i$ averages over the N future targets and sg denotes stop gradient. The joint objective is $\mathcal{L}_{\text{LeWAM}} = \mathcal{L}_{\text{act}} + \alpha\mathcal{L}_{\text{wm}}$, where α controls the contribution of future embedding prediction.

Inference. At inference, LeWAM encodes X_t into Z_0 and denoises an action chunk of the desired length conditioned on Z_0 and c_t. Clean action tokens and future queries are omitted, so future embedding prediction provides training supervision without additional inference cost.

3.4 Demonstration-Guided DPO

We observe that LeWAM can generate action chunks that closely match successful trajectories, yet small action deviations can still cause a rollout to fail. This exposes a limitation of supervised imitation learning: flow matching learns the demonstrated action distribution but provides no explicit signal that distinguishes better generated actions from worse ones. We therefore refine LeWAM with Demonstration-Guided DPO (DemoDPO), which uses each demonstrated action chunk to rank a group of candidates sampled from a frozen reference policy. We initialize the trainable policy π_θ and a frozen reference policy π_{ref} from the supervised LeWAM checkpoint. For each training condition $x_t = (X_t, c_t)$ and demonstrated action chunk A_t, we draw G candidates $\widetilde{A}_i \overset{\text{i.i.d.}}{\sim} \pi_{\text{ref}}(\cdot \mid x_t)$ from independent Gaussian noise. We measure their similarity to the demonstration using action MSE, denoted by d_i, and define the candidates with the smallest and largest MSE as A^w and A^l, respectively. The demonstration therefore provides the ranking signal, while both candidates are sampled from the same frozen reference policy. Let i^w and i^l be the indices of A^w and A^l, respectively, and let $\varepsilon_{\text{pair}}$ denote the minimum MSE gap. We define the pair retention mask as

$$M_{\text{pair}} = \mathbf{1}\left\{d_{i^l} - d_{i^w} \geq \varepsilon_{\text{pair}}\right\}. \tag{7}$$

We retain only pairs with $M_{\text{pair}} = 1$, filtering out groups whose best and worst candidates have nearly indistinguishable demonstration errors.

For a preference pair (A^w, A^l), standard DPO (Rafailov et al., 2023) minimizes

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log\sigma\left(\beta\left[\rho_\theta(A^w \mid x_t) - \rho_\theta(A^l \mid x_t)\right]\right)\right], \qquad \rho_\theta(A \mid x_t) = \log\frac{\pi_\theta(A \mid x_t)}{\pi_{\text{ref}}(A \mid x_t)}, \tag{8}$$

where the expectation is over preference pairs, σ is the sigmoid function, β is the DPO regularization coefficient, and $\rho_\theta(A \mid x_t)$ is the log policy ratio of the trainable policy to the frozen reference policy. A larger $\rho_\theta(A \mid x_t)$ means that the trainable policy assigns higher likelihood to A relative to the reference policy. For a flow matching policy, ρ_θ cannot be evaluated directly from the velocity predictor. Diffusion-DPO (Wallace et al., 2024) derives a tractable surrogate objective for diffusion models from denoising errors at a sampled noise level, while Flow-DPO (Liu et al., 2026) adapts this objective to flow matching policies using velocity prediction errors. We follow this construction by sampling (τ, ϵ) independently of the candidate generation noises $\{\eta_i\}_{i=1}^G$ and applying Equation 5 to A^w and A^l with the same (τ, ϵ). For $\phi \in \{\theta, \text{ref}\}$ and $A \in \{A^w, A^l\}$, let A'_τ denote the resulting noisy action. We define the mean squared flow matching error as

$$\bar{\ell}_\phi(A \mid x_t; \tau, \epsilon) = \frac{1}{H_a D_a}\left\|\hat{v}_\phi(A'_\tau, x_t, \tau) - (\epsilon - A)\right\|_2^2, \tag{9}$$

where D_a is the action dimension. The corresponding tractable surrogate for the log policy ratio is

$$s_\theta(A \mid x_t; \tau, \epsilon) = -\frac{1}{2}\left[\bar{\ell}_\theta(A \mid x_t; \tau, \epsilon) - \bar{\ell}_{\text{ref}}(A \mid x_t; \tau, \epsilon)\right]. \tag{10}$$

Substituting the surrogate s_θ for ρ_θ in Equation 8 gives

$$\mathcal{L}_{\text{DemoDPO}} = -\mathbb{E}\left[\log\sigma\left(\beta\left[s_\theta(A^w \mid x_t;\tau,\epsilon) - s_\theta(A^l \mid x_t;\tau,\epsilon)\right]\right) \mid M_{\text{pair}} = 1\right]. \quad (11)$$

The expectation is over candidate groups sampled from the frozen reference policy and the shared flow noise, conditioned on $M_{\text{pair}} = 1$. The loss encourages the trainable policy to assign a higher relative score to the candidate with lower demonstration MSE than to the candidate with higher demonstration MSE. Appendix B.1 derives this objective from the Diffusion-DPO denoising surrogate and the Flow-DPO velocity formulation, and Algorithm 1 summarizes one update step.

4 Experiments

4.1 Experimental Setup

Simulation and real-world settings. We evaluate LeWAM on both simulated and real-world manipulation tasks. For simulation, we use RoboTwin 2.0, a bimanual robotic manipulation benchmark comprising 50 tasks, following the multitask training setup of recent RoboTwin evaluations (Ye et al., 2026a; Yuan et al., 2026). The training data contain 2,500 clean and 25,000 heavily randomized demonstrations, corresponding to 50 and 500 demonstrations per task, respectively. We train all policies in our controlled comparisons for 10 epochs using AdamW with a learning rate of 10^{-4} and a global batch size of 1024 on 16 NVIDIA H800 GPUs. Following Fast-WAM, we use an action chunk of 32 steps in simulation. For future embedding prediction, we uniformly sample one future offset from steps 1 to 32 for each training sample and set $\alpha = 0.1$. For DemoDPO, we sample $G = 4$ candidates from the frozen reference policy using 10 explicit Euler steps and retain pairs satisfying $d_{i^l} - d_{i^w} \geq 10^{-3}$. We train DemoDPO for 1,000 optimization steps with a learning rate of 10^{-5}, $\beta = 5000$, and an EMA decay of 0.9995. In the real world, we evaluate three bimanual tasks on an AgileX dual Piper robot: stacking three blocks, folding a towel, and arranging three flowers in a vase.

Evaluation protocol. Simulation performance is measured by task success rate. For the real-world evaluation, each method is evaluated for 50 rollouts per task, and we report task progress measured by milestone completion. The complete real-world experiment details are provided in Appendix D. Table 3 in Appendix C.4 reports inference latency and peak memory.

4.2 RoboTwin 2.0 Results and Ablations

We organize the RoboTwin 2.0 evaluation around three empirical questions: representation choice, latent world modeling, and preference refinement.

- *RQ1*: How do frozen visual representations affect action generation?
- *RQ2*: How should future embeddings be predicted in JEPA space?
- *RQ3*: What makes offline preference refinement effective for LeWAM?

***RQ1*: Frozen visual representations.** Proposition 1 and the decomposition in Appendix A.3 attribute action uncertainty to the information a representation omits and the information a fixed predictor cannot extract, so we compare frozen vision encoders under the same action predictor and training recipe. We use a discrete task ID as c_t across all controlled comparisons to isolate visual representation effects from differences in text representations. We first compare the Wan2.2 VAE, SigLIP, MAE-Large, DINOv3-Large, and V-JEPA2-Large, as shown in Figure 1*(a)* and Table 1. The Wan2.2 VAE performs worst at 10.81%, while V-JEPA2-Large performs best among the three Large encoders. Notably, I-JEPA-Huge outperforms V-JEPA2-Huge despite being pretrained only on ImageNet-22K (Deng et al., 2009).

To further understand why I-JEPA supports action generation effectively, we visualize the attention from action tokens to vision tokens during inference. Figure 4 shows that the attention concentrates on the target objects and target regions and shifts between relevant regions as the manipulation progresses. Prior work links segmentation ability to manipulation generalization (Burns et al., 2023) and shows benefits of geometric information for robot learning (Jeon et al., 2026; Wang et al., 2024), so we examine how readily each frozen encoder exposes geometry and object information. We train lightweight depth and segmentation probes on the frozen encoder outputs with teacher targets from

DA3MONO-LARGE (Lin et al., 2025) and SAM3 (Carion et al., 2026), following the protocol in Appendix C.5. I-JEPA-Huge performs best among the evaluated individual encoders on both probes while the Wan2.2 VAE performs worst, indicating that I-JEPA makes the geometry and object information emphasized during action generation readily accessible.

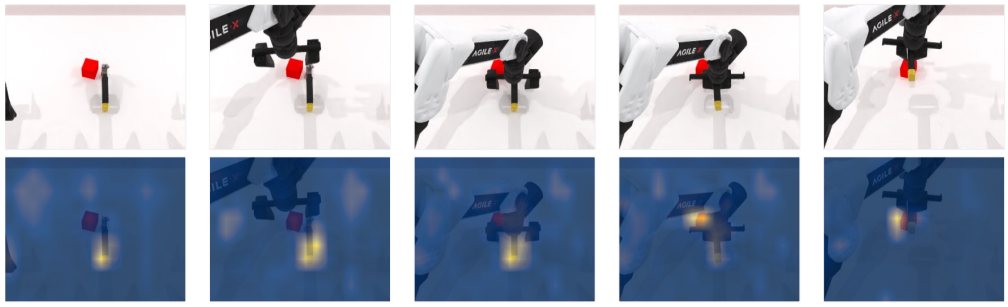


Figure 4: Attention from action tokens to vision tokens during a representative hammering rollout.

Table 1: Frozen encoder comparison on RoboTwin 2.0 with representation probing. Bold, underline, and italics mark the best, second, and third results, respectively.

	Success rate			Depth		Segmentation	
Encoder	Clean	Rand.	Avg.	RMSE ↓	Corr. ↑	AP@50 ↑	AP@75 ↑
Wan2.2-VAE	12.52	9.10	10.81	0.708	0.600	0.206	0.025
SigLIP	81.60	76.80	79.20	0.625	0.711	0.378	0.044
MAE-Large	64.30	63.80	64.05	0.556	0.774	0.535	0.147
DINOv3-Large	69.58	65.50	67.54	0.572	0.765	0.515	0.146
V-JEPA2-Large	73.40	70.94	72.17	0.477	0.832	*0.571*	*0.247*
V-JEPA2-Huge	*83.76*	*79.64*	*81.70*	*0.446*	*0.852*	0.532	0.238
I-JEPA-Huge	<u>88.42</u>	<u>83.92</u>	<u>86.17</u>	<u>0.445</u>	<u>0.853</u>	<u>0.613</u>	<u>0.323</u>
+ AdaFuse	**88.76**	**85.00**	**86.88**	**0.440**	**0.854**	**0.616**	**0.345**

Table 2: RoboTwin baselines and LeWAM variants. We report average success rates over 50 tasks in clean and randomized settings, with 100 evaluation cases per task in each setting. Embodied PT. indicates whether additional robot data are used for pretraining before benchmark training.

Model	Encoder	Trainable Params.	Embodied PT.	Clean	Rand.	Avg.
VLM-based VLAs						
π_0	SigLIP	3B	✓	65.92	58.40	62.16
$\pi_{0.5}$	SigLIP	3B	✓	82.74	76.76	79.75
VDM-based WAMs						
Motus w/o Pretrain	Wan2.2-VAE	6B	✗	77.56	77.00	77.28
LingBot-VA w/o Pretrain	Wan2.2-VAE	5B	✗	80.60	–	80.60
GigaWorld-Policy	Wan2.2-VAE	5B	✓	86.36	85.04	85.70
Motus	Wan2.2-VAE	6B	✓	88.66	87.02	87.84
Fast-WAM	Wan2.2-VAE	6B	✗	91.88	91.78	91.83
LingBot-VA	Wan2.2-VAE	5B	✓	92.90	91.50	92.20
Latent-based WAMs						
LaWAM	DINOv3-B/16	1.7B	✓	92.64	89.80	91.22
LeWAM Variants						
Action Only	I-JEPA-H/14	0.4B	✗	88.42	83.92	86.17
+ WM (FM)	I-JEPA-H/14	0.4B	✗	88.18	84.24	86.21
+ WM	I-JEPA-H/14	0.4B	✗	88.86	85.18	87.02
+ WM + AdaFuse	I-JEPA-H/14	0.4B	✗	91.32	90.06	90.69
LeWAM	I-JEPA-H/14	0.4B	✗	**93.14**	**91.42**	**92.28**

***RQ2*: Future embedding prediction.** We keep LeWAM's I-JEPA encoder frozen and compare direct prediction of future embeddings from learnable future queries, as described in Section 3.3, with a flow matching alternative. The latter adds Gaussian noise to future embeddings and predicts their transport directions conditioned on the corresponding clean action prefixes.

Table 2 shows that direct prediction achieves 87.02%, outperforming flow matching at 86.21% and the Action Only baseline at 86.17%. Prior work shows that JEPA representations encode rich se-

mantic and spatial structure (Assran et al., 2023; Mur-Labadia et al., 2026). In our setting, the frozen encoder maps each observed future frame to a fixed, high-dimensional target embedding, whereas flow matching introduces an additional noise-conditioned transport problem over this space. We hypothesize that direct regression makes more direct use of this structured supervision, explaining its stronger performance. We then evaluate AdaFuse, which adaptively integrates representations from multiple encoder layers. Without future embedding prediction, AdaFuse improves success from 86.17% to 86.88%, as shown in Table 1. Adding future embedding prediction further raises success to 90.69% in Table 2, with a larger improvement than without AdaFuse. These results suggest that AdaFuse and future embedding prediction are complementary. Table 1 further shows that AdaFuse improves depth and segmentation probing over the final I-JEPA layer, suggesting that the learned fusion makes complementary information across encoder layers more accessible.

***RQ3*: DemoDPO refinement.** DemoDPO improves the average success rate from 90.69% to 92.28% across all 50 RoboTwin tasks without additional environment interaction, as shown in Table 2. To examine what makes this refinement effective, Figure 5 compares pair filtering thresholds, DPO coefficients, and candidate group sizes on a fixed subset of eight challenging tasks. The evaluation protocol is provided in Appendix C.3. Pair filtering retains candidates with a sufficient difference in demonstration similarity. A moderate threshold gives the strongest improvement, whereas using all pairs or imposing an overly strict threshold reduces performance. This suggests that effective refinement requires both distinguishable preferences and sufficient training pairs. The DPO coefficient also affects refinement: performance improves as β increases to 5000, then declines at 6000. Increasing the candidate group size from 2 to 4 improves success, while increasing it further to 8 provides little additional benefit. We therefore use $\varepsilon_{\text{pair}} = 10^{-3}$, $\beta = 5000$, and $G = 4$.

To distinguish the benefit of DemoDPO from that of additional training, we apply 1,000 additional supervised updates to the checkpoint used to initialize DemoDPO. Success decreases slightly from 73.31% to 73.25%, whereas DemoDPO reaches 76.63% after the same number of updates, indicating that its gains are not explained by additional training alone. LeWAM therefore matches the strongest baselines in Table 2 with 0.4B trainable parameters and no video diffusion backbone, at the lowest latency and peak memory among the evaluated models in Table 3.

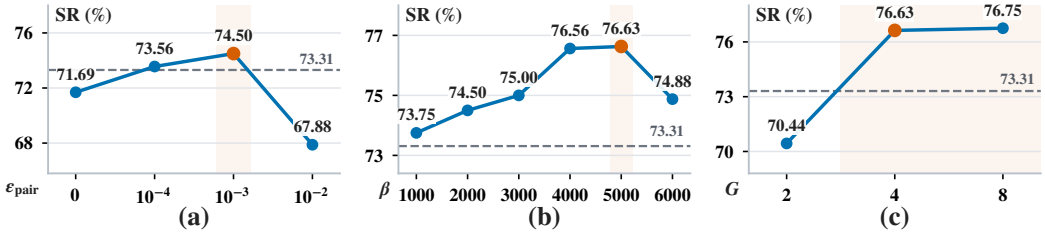


Figure 5: DemoDPO analyses and ablations for RQ3. *(a)* Pair filtering threshold. *(b)* DPO coefficient. *(c)* Candidate group size. The dashed line marks the 73.31% success rate before DemoDPO.

Real-world experiments. Figure 1*(b)* summarizes the real-world results, with further results and implementation details provided in Appendix D.

5 Conclusion

Our theoretical analysis motivates examining both the information retained by visual representations and its accessibility for action generation. Attention analysis shows that the policy focuses on target objects and regions during manipulation, while controlled encoder comparisons and representation probing suggest that I-JEPA makes the relevant geometric and object information more accessible than raw VAE representations. These findings motivate LeWAM, which couples action generation with future embedding prediction in I-JEPA space rather than relying on a pretrained video diffusion backbone. AdaFuse strengthens this representation by integrating complementary information across encoder layers, while DemoDPO further refines the policy using preferences derived from demonstrations. Together, these components enable LeWAM to achieve an average success rate of 92.28% on RoboTwin 2.0 and strong performance on real-world manipulation tasks, with only 0.4B trainable parameters and no embodied pretraining.

AI USE STATEMENT

In this work, we used generative AI tools to assist with codebase development and manuscript polishing. Generative AI was not used to produce conclusions unsupported by objective experimental evidence. All AI-assisted content, including but not limited to code and manuscript text, was manually reviewed and verified by the authors. We take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide the training and inference code in the supplementary material. Our simulation experiments use the official RoboTwin 2.0 training dataset. The real-world experiments use an in-house dataset collected through teleoperation, which we will publicly release. Detailed model configurations, training procedures, and evaluation protocols are provided in the paper and appendix.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

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.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. RT-H: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Kaylee Burns, Zach Witzel, Jubayer Ibn Hamid, Tianhe Yu, Chelsea Finn, and Karol Hausman. What makes pre-trained visual representations successful for robust manipulation? *arXiv preprint arXiv:2312.12444*, 2023.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. SAM 3: Segment anything with concepts. In *International Conference on Learning Representations*, volume 2026, pp. 138846–138923, 2026.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

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

Peng Chen, Pi Bu, Yingyao Wang, Xinyi Wang, Ziming Wang, Jie Guo, Yingxiu Zhao, Qi Zhu, Jun Song, Siran Yang, et al. CombatVLA: An efficient vision-language-action model for combat tasks in 3D action role-playing games. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10919–10928, 2025a.

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *International Journal of Computer Vision*, 132(1):208–223, 2024.

Yuxin Chen, Devesh K Jha, Masayoshi Tomizuka, and Diego Romeres. FDPP: Fine-tune diffusion policy with human preference. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12010–12016. IEEE, 2025b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

ThankGod Egbe, Peng Wang, Zhihao Guo, and Zidong Chen. DINOv3-diffusion policy: Self-supervised large visual model for visuomotor diffusion policy learning. *arXiv preprint arXiv:2509.17684*, 2025.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Yiguo Fan, Pengxiang Ding, Shuanghao Bai, Xinyang Tong, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, et al. Long-VLA: Unleashing long-horizon capability of vision language action model for robot manipulation. *arXiv preprint arXiv:2508.19958*, 2025.

GigaBrain Team, Boyuan Wang, Bohan Li, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. GigaBrain-0.5M*: a VLA that learns from world model-based reinforcement learning. *arXiv preprint arXiv:2602.12099*, 2026.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

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. *arXiv preprint arXiv:2412.14803*, 2024.

Byungwoo Jeon, Dongyoung Kim, Huiwon Jang, Insoo Kim, and Jinwoo Shin. SpatialBoost: Enhancing visual representation through language-guided reasoning. *arXiv preprint arXiv:2603.22057*, 2026.

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

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

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, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jie Liu, Gongye Liu, Jiajun Liang, Ziyang Yuan, Xiaokun Liu, Mingwu Zheng, Xiele Wu, Qiulin Wang, Menghan Xia, Xintao Wang, et al. Improving video generation with human feedback. *Advances in Neural Information Processing Systems*, 38:82155–82192, 2026.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. DiT4DiT: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorld-Model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Shangchen Miao, Ningya Feng, Jialong Wu, Ye Lin, Xu He, Dong Li, and Mingsheng Long. JEPA-VLA: Video predictive embedding is needed for VLA models. *arXiv preprint arXiv:2602.11832*, 2026.

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.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Darshan Patil, Janarthanan Rajendran, Glen Berseth, and Sarath Chandar. Intelligent switching for reset-free RL. *arXiv preprint arXiv:2405.01684*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 4172–4182. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00387. URL https://doi.org/10.1109/ICCV51070.2023.00387.

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.

Guo-Jun Qi, Liheng Zhang, Chang Wen Chen, and Qi Tian. AVT: Unsupervised learning of transformation equivariant representations by autoencoding variational transformations. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8129–8138. IEEE, 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Zhao Shan, Chenyou Fan, Shuang Qiu, Jiyuan Shi, and Chenjia Bai. Forward KL regularized preference optimization for aligning diffusion policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14386–14395, 2025.

Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Formalism and benchmarking. *arXiv preprint arXiv:2112.09605*, 2021.

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. *arXiv preprint arXiv:2602.10098*, 2026.

Amitesh Vatsa, Zhixian Xie, and Wanxin Jin. RoDiF: Robust direct fine-tuning of diffusion policies with corrupted human feedback. *arXiv preprint arXiv:2602.00886*, 2026.

Homer Rich Walke, Jonathan Heewon Yang, Albert Yu, Aviral Kumar, Jedrzej Orbik, Avi Singh, and Sergey Levine. Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning. In *Conference on Robot Learning*, pp. 1652–1662. PMLR, 2023.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Letian Wang, Chuhan Zhang, Rishabh Kabra, Jasper Uijlings, Steven Waslander, Andrew Zisserman, Joao Carreira, Kaiming He, Misha Andriluka, Eduard Gabriel Bazavan, et al. Video generation models are general-purpose vision learners. *arXiv preprint arXiv:2607.09024*, 2026.

Wanying Wang, Jinming Li, Yichen Zhu, Zhiyuan Xu, Zhengping Che, Yaxin Peng, Chaomin Shen, Dong Liu, Feifei Feng, and Jian Tang. Visual robotic manipulation with depth-aware pretraining. *arXiv preprint arXiv:2401.09038*, 2024.

Zihao Wang and Liu Ziyin. Posterior collapse of a linear latent variable model. *Advances in Neural Information Processing Systems*, 35:37537–37548, 2022.

Ziru Wang, Mengmeng Wang, Jade Dai, Teli Ma, Guo-Jun Qi, Yong Liu, Guang Dai, and Jingdong Wang. DynaMind: Reasoning over abstract video dynamics for embodied decision-making. In *Forty-second International Conference on Machine Learning*, 2025.

Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. LLaDA-VLA: Vision language diffusion action models. *arXiv preprint arXiv:2509.06932*, 2025.

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 *International Conference on Learning Representations*, volume 2024, pp. 10641–10662, 2024.

Wenke Xia, Yichu Yang, Hongtao Wu, Xiao Ma, Tao Kong, and Di Hu. Human-assisted robotic policy refinement via action preference optimization. *Advances in Neural Information Processing Systems*, 38:36746–36768, 2026.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=r1eBeyHFDH`.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. GigaWorld-Policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a.

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

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, et al. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026.

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

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. AET vs. AED: Unsupervised representation learning by auto-encoding transformations rather than data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2542–2550. IEEE, 2019.

Tianle Zhang, Jiayi Guan, Lin Zhao, Yihang Li, Dongjiang Li, Zecui Zeng, Lei Sun, Yue Chen, Xuelong Wei, Lusong Li, et al. Preferred-action-optimized diffusion policies for offline reinforcement learning. *arXiv preprint arXiv:2405.18729*, 2024a.

Xinyu Zhang, Jiahui Chen, Junkun Yuan, Qiang Chen, Jian Wang, Xiaodi Wang, Shumin Han, Xiaokang Chen, Jimin Pi, Kun Yao, et al. CAE v2: Context autoencoder with CLIP latent alignment. *Transactions on Machine Learning Research*, 2023.

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. GRAPE: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024b.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

Han Zhou, Jinjin Cao, Liyuan Ma, Xueji Fang, and Guo-Jun Qi. From human hands to robot arms: Manipulation skills transfer via trajectory alignment. *CoRR*, abs/2510.00491, 2025. doi: 10.48550/ARXIV.2510.00491. URL https://doi.org/10.48550/arXiv.2510.00491.

Jie Zhu, Jiyang Qi, Mingyu Ding, Xiaokang Chen, Ping Luo, Xinggang Wang, Wenyu Liu, Leye Wang, and Jingdong Wang. Understanding self-supervised pretraining with part-aware representation learning. *arXiv preprint arXiv:2301.11915*, 2023.

APPENDIX

A THEORETICAL ANALYSIS

This section proves Propositions 1 and 2 and analyzes the effective dimension and action information of VAE posterior means.

A.1 ACTION-RELEVANT INFORMATION BOUNDS

We use the same notation as the main text: X is the current visual observation, S denotes the action-relevant explanatory factors, A is the expert action chunk, c is task context, and $Z = E(X)$ is the visual representation consumed by the action predictor.

Proof of Proposition 1. Let $Z = E(X)$ and suppose $H(A \mid Z, c)$ and $H(A \mid X, c)$ are finite. Because Z is determined by X, conditioning on (X, Z, c) is equivalent to conditioning on (X, c).

$$I(A; X \mid Z, c) = H(A \mid Z, c) - H(A \mid X, Z, c) = H(A \mid Z, c) - H(A \mid X, c) = \Delta_E(A). \quad (12)$$

Since $S = S(X)$, the mutual information chain rule gives

$$I(A; X \mid Z, c) = I(A; S, X \mid Z, c) = I(A; S \mid Z, c) + I(A; X \mid S, Z, c). \quad (13)$$

Assumption 1 states that A is conditionally independent of X given (S, c). Since Z is a function of X, this also gives $I(A; X \mid S, Z, c) = 0$. Combining the two identities and using nonnegativity of conditional mutual information yields

$$\Delta_E(A) = I(A; S \mid Z, c) \geq 0. \quad (14)$$

This identity motivates evaluating a representation by the action-relevant factor information it preserves rather than by its image reconstruction quality.

A.2 SPECTRAL COMPRESSION OF VAE POSTERIOR MEANS

We analyze how VAE training selects the effective dimension of the posterior mean, then determine when the removed directions contain action information. The spectral analysis specializes the linear VAE result of Wang & Ziyin (2022). The action uncertainty result below follows by conditioning on the retained directions. This analysis concerns a linear Gaussian model at a global optimum, with a fixed decoder variance. The linear Gaussian model is an analyzable instance, not a model of the pretrained Wan2.2 VAE, whose encoder is nonlinear and whose pretraining distribution differs from the demonstration distribution. We use it to establish what reconstruction based selection does and does not guarantee about action information, and we treat the pretrained encoder empirically in Section 4.2.

Fix a task context c and let the centered observation vector be $X \in \mathbb{R}^{d_x}$ with

$$X \mid c \sim \mathcal{N}(0, \Sigma_X), \qquad \Sigma_X = \sum_{i=1}^{d_x} \lambda_i u_i u_i^\top, \qquad \lambda_1 > \cdots > \lambda_{d_x} > 0. \quad (15)$$

The u_i are orthonormal principal directions, and λ_i measures observation variance along u_i. Consider a VAE with $k \leq d_x$ latent coordinates, standard normal prior, and

$$q(\tilde{z} \mid x) = \mathcal{N}(Mx, \mathrm{diag}(s_1^2, \ldots, s_k^2)), \qquad p(x \mid \tilde{z}) = \mathcal{N}(W\tilde{z}, \sigma_{\mathrm{dec}}^2 \mathbf{I}), \quad (16)$$

where $M \in \mathbb{R}^{k \times d_x}$, $W \in \mathbb{R}^{d_x \times k}$, and $s_i^2 > 0$ are learned, while $\sigma_{\mathrm{dec}}^2 > 0$ is fixed. The encoder variances are independent of x. Let $\widetilde{Z}$ denote the sampled latent variable and $\tilde{z}$ its realization. The training objective is, up to a constant,

$$\mathcal{L}_{\mathrm{vae}} = \frac{1}{2\sigma_{\mathrm{dec}}^2} \mathbb{E}_{X|c} \mathbb{E}_{\widetilde{Z}|X} \left[\|X - W\widetilde{Z}\|_2^2 \right] + \beta_{\mathrm{vae}} \, \mathbb{E}_{X|c} D_{\mathrm{KL}}\big(q(\tilde{z} \mid X) \| \mathcal{N}(0, \mathbf{I})\big). \quad (17)$$

Here $\beta_{\mathrm{vae}} > 0$ is the VAE regularization weight. Although training samples $\widetilde{Z}$, the representation supplied to the action predictor is the deterministic posterior mean $Z_{\mathrm{vae}} = \mathbb{E}[\widetilde{Z} \mid X] = MX$. Let $\lambda_{\mathrm{cut}} = \beta_{\mathrm{vae}} \sigma_{\mathrm{dec}}^2$. At a global optimum expressed in principal coordinates, the posterior mean has coordinates

$$(Z_{\mathrm{vae}})_i = \frac{\sqrt{(\lambda_i - \lambda_{\mathrm{cut}})_+}}{\lambda_i} u_i^\top X, \qquad i = 1, \ldots, k, \quad (18)$$

where $(a)_+ = \max(a, 0)$. Its effective dimension, defined as the rank of its covariance, is

$$r_{\text{eff}} := \operatorname{rank} \operatorname{Cov}(Z_{\text{vae}} \mid c) = \min\big(k, \#\{i : \lambda_i > \lambda_{\text{cut}}\}\big). \tag{19}$$

Let J be the index set of retained directions, with $J = \{i \le k : \lambda_i > \lambda_{\text{cut}}\}$, and define $\Sigma_{\text{omit}} = \sum_{i \notin J} \lambda_i u_i u_i^\top$. If the expert action chunk, viewed as a vector, satisfies

$$A = B_c X + \xi, \qquad \xi \sim \mathcal{N}(0, \Sigma_\xi), \qquad \xi \perp X \mid c, \qquad \Sigma_\xi \succ 0, \tag{20}$$

then the additional action uncertainty is

$$\Delta_E(A) = I(A; X \mid Z_{\text{vae}}, c) = \frac{1}{2} \log\det\left(\mathbf{I} + \Sigma_\xi^{-1/2} B_c \Sigma_{\text{omit}} B_c^\top \Sigma_\xi^{-1/2}\right). \tag{21}$$

It is strictly positive exactly when $B_c u_i \neq 0$ for at least one omitted direction.

Proof of Proposition 2. **1. Reduce training to principal directions.** The spectral solution for the linear VAE aligns the decoder with principal directions of Σ_X (Wang & Ziyin, 2022, Theorem 2). For completeness, we derive the resulting threshold and its effect on the posterior mean. Write $x_i = u_i^\top X$, and denote the scalar decoder loading, encoder mean coefficient, and posterior variance along this direction by w_i, m_i, and s_i^2. The contribution of this direction to Equation 17 is

$$\ell_i = \frac{\lambda_i (1 - w_i m_i)^2 + w_i^2 s_i^2}{2\sigma_{\text{dec}}^2} + \frac{\beta_{\text{vae}}}{2}\left(\lambda_i m_i^2 + s_i^2 - \log s_i^2 - 1\right). \tag{22}$$

The first term rewards reconstruction, while the second penalizes the encoder mean and deviation of the posterior variance from the prior variance.

2. Identify which directions survive regularization. Fix w_i. Minimizing over m_i and s_i^2 gives

$$m_i^* = \frac{w_i}{w_i^2 + \lambda_{\text{cut}}}, \qquad (s_i^2)^* = \frac{\lambda_{\text{cut}}}{w_i^2 + \lambda_{\text{cut}}}. \tag{23}$$

Substituting these values and writing $t_i = w_i^2 \ge 0$ reduces the objective to

$$\ell_i^*(t_i) = \frac{\beta_{\text{vae}}}{2}\left[\frac{\lambda_i}{t_i + \lambda_{\text{cut}}} + \log\left(1 + \frac{t_i}{\lambda_{\text{cut}}}\right)\right], \qquad \frac{d\ell_i^*}{dt_i} = \frac{\beta_{\text{vae}}}{2} \frac{t_i + \lambda_{\text{cut}} - \lambda_i}{(t_i + \lambda_{\text{cut}})^2}. \tag{24}$$

Thus $(w_i^2)^* = (\lambda_i - \lambda_{\text{cut}})_+$. Choosing nonnegative decoder loadings gives Equation 18. Sign changes or permutations of latent coordinates do not affect the result. When $\lambda_i \le \lambda_{\text{cut}}$, both w_i^* and m_i^* vanish, and $(s_i^2)^* = 1$: this coordinate matches the prior and its mean carries no observation information. When $\lambda_i > \lambda_{\text{cut}}$, its mean coefficient is nonzero. The reduction in the optimized loss relative to omitting a direction increases with λ_i above the threshold, so the k available coordinates retain the largest eligible eigenvalues. Their number is given by Equation 19.

3. Determine the information missing from the posterior mean. For every $i \in J$, the nonzero coefficient in Equation 18 makes $u_i^\top X$ recoverable from Z_{vae}. For $i \notin J$, the corresponding coordinate is absent. The Gaussian principal coordinates are independent, so

$$\mathbb{E}[X \mid Z_{\text{vae}}, c] = \sum_{i \in J} u_i u_i^\top X, \qquad \operatorname{Cov}(X \mid Z_{\text{vae}}, c) = \Sigma_{\text{omit}}. \tag{25}$$

This identifies the lost directions for the deterministic mean, rather than a noisy latent sample.

4. Translate missing observation directions into action uncertainty. Applying Equation 20 gives

$$\operatorname{Cov}(A \mid X, c) = \Sigma_\xi, \qquad \operatorname{Cov}(A \mid Z_{\text{vae}}, c) = \Sigma_\xi + B_c \Sigma_{\text{omit}} B_c^\top. \tag{26}$$

Both conditional action distributions are Gaussian with positive definite covariance. Their differential entropies are finite, and subtracting them yields

$$\Delta_E(A) = H(A \mid Z_{\text{vae}}, c) - H(A \mid X, c) = \frac{1}{2} \log \frac{\det(\Sigma_\xi + B_c \Sigma_{\text{omit}} B_c^\top)}{\det \Sigma_\xi}, \tag{27}$$

which equals the expression in Equation 21. Here H denotes differential entropy. No nonnegativity assumption on differential entropy is used. Finally, $B_c \Sigma_{\text{omit}} B_c^\top = \sum_{i \notin J} \lambda_i (B_c u_i)(B_c u_i)^\top$ is positive semidefinite and is nonzero exactly when an omitted direction affects the action. The log determinant gap is therefore strictly positive precisely under the stated condition.

Corollary 1 (Reconstruction ordering does not order action relevance)**.** *In the setting of Proposition 2 with $k < d_x$, let $B_c = b\, u_{d_x}^\top$, so the action depends only on the least energetic observation direction. Then $\Delta_E(A) = I(A; X \mid c)$, that is, Z_{vae} retains no action information, whereas the single direction $u_{d_x}^\top X$ attains $\Delta_E(A) = 0$ while having the largest reconstruction error among principal directions. There exist sequences of observation and action noise covariances for which the minimum reconstruction error from Z_{vae} tends to zero while $\Delta_E(A)$ diverges.*

Proof of Corollary 1. Since $k < d_x$, the index d_x is omitted, so $\lambda_{d_x} u_{d_x} u_{d_x}^\top \preceq \Sigma_{\text{omit}}$ and $B_c \Sigma_{\text{omit}} B_c^\top = \lambda_{d_x} b b^\top = B_c \Sigma_X B_c^\top$, using $B_c = b\, u_{d_x}^\top$ and $u_{d_x}^\top \Sigma_X u_{d_x} = \lambda_{d_x}$. Substituting into Equation 21 and applying $\det(\mathbf{I} + vv^\top) = 1 + \|v\|_2^2$ gives

$$\Delta_E(A) = \frac{1}{2} \log\bigl(1 + \lambda_{d_x} b^\top \Sigma_\xi^{-1} b\bigr) = \frac{1}{2} \log \frac{\det(\Sigma_\xi + B_c \Sigma_X B_c^\top)}{\det \Sigma_\xi} = I(A; X \mid c), \tag{28}$$

so the posterior mean retains none of the action information in X. For the representation $u_{d_x}^\top X$, the action is conditionally independent of X given $(u_{d_x}^\top X, c)$, so $\Delta_E(A) = 0$, while its reconstruction error $\operatorname{tr} \Sigma_X - \lambda_{d_x}$ is the largest among projections onto a single principal direction. Fix $\lambda_1 > \cdots > \lambda_k > \lambda_{\text{cut}}$ and take $0 < \epsilon < \lambda_{\text{cut}}$. Choose distinct $\lambda_i \in (\epsilon/2, \epsilon]$ for all $i > k$, together with $\Sigma_\xi = \epsilon^2 \mathbf{I}$ and $\|b\|_2 = 1$. Then $J = \{1, \ldots, k\}$. Then $\operatorname{tr} \Sigma_{\text{omit}} \le (d_x - k)\epsilon \to 0$ while $\lambda_{d_x} b^\top \Sigma_\xi^{-1} b \ge 1/(2\epsilon) \to \infty$.

The result separates two sources of dimensional reduction: the architectural width k and the threshold $\beta_{\text{vae}} \sigma_{\text{dec}}^2$ induced by training. Within this model, directions are retained according to their contribution to visual reconstruction, while their effect on action uncertainty depends additionally on B_c. The effective dimension can be smaller than the latent width, and losing a direction increases action uncertainty only when that direction matters for the task. Corollary 1 and Proposition 3 bound the two ends of this behavior: reconstruction based selection can retain no action information when the action depends on a low variance direction, and it can lose only a controlled amount when reconstruction is accurate and the action sensitivity is bounded.

A.3 Usable Action Uncertainty

Proposition 1 measures information content and is indifferent to whether a downstream model can extract it. We record here the decomposition used in Section 3.2 and the bound that separates its two terms. Let $\mathcal{V}$ be a family of conditional densities for A given (Z, c) and let $H_{\mathcal{V}}(A \mid Z, c) = \inf_{f \in \mathcal{V}} \mathbb{E}[-\log f(A \mid Z, c)]$ be the usable conditional entropy of Xu et al. (2020). For any f, $\mathbb{E}[-\log f(A \mid Z, c)] - H(A \mid Z, c) = \mathbb{E}_{Z,c} D_{\text{KL}}\bigl(p(\cdot \mid Z, c) \,\|\, f(\cdot \mid Z, c)\bigr) \ge 0$, so $H_{\mathcal{V}}(A \mid Z, c) \ge H(A \mid Z, c)$, with equality when $\mathcal{V}$ contains the true conditional. Adding and subtracting $H(A \mid Z, c)$ yields

$$\underbrace{H_{\mathcal{V}}(A \mid Z, c) - H(A \mid X, c)}_{\text{optimal excess log loss}} = \underbrace{\Delta_E(A)}_{\text{information loss}} + \underbrace{H_{\mathcal{V}}(A \mid Z, c) - H(A \mid Z, c)}_{\text{usability deficit}}, \tag{29}$$

whose first term is nonnegative by Proposition 1 and whose second term is nonnegative by the inequality above. The first term depends only on the representation, while the second depends jointly on the representation and the downstream model. Here A denotes the vectorized action chunk of dimension $H_a D_a$. If $\mathcal{V}$ consists of predictors $\mathcal{N}(g(Z, c), \sigma^2 \mathbf{I})$ with g ranging over the realizable regressors and $\sigma^2 > 0$ free, and $0 < \overline{M}(Z) < \infty$, then taking the infimum over g and optimizing σ^2 gives $\sigma^2 = \overline{M}(Z)/(H_a D_a)$ and

$$H_{\mathcal{V}}(A \mid Z, c) = \frac{H_a D_a}{2} \log\left(\frac{2\pi e\, \overline{M}(Z)}{H_a D_a}\right), \qquad \overline{M}(Z) = \inf_g \mathbb{E}\left[\|A - g(Z, c)\|_2^2\right]. \tag{30}$$

For this Gaussian family, usable conditional entropy is monotone in the minimum achievable action MSE.

Proposition 3 (Loss channel bound)**.** *In the setting of Proposition 2 with $A = B_c X + \xi$ as in Equation 20,*

$$\Delta_E(A) \le \frac{H_a D_a}{2} \log\left(1 + \frac{\|B_c\|_2^2 \,\text{MSE}_{\text{rec}}}{\lambda_{\min}(\Sigma_\xi)}\right), \quad \text{MSE}_{\text{rec}} = \operatorname{tr} \Sigma_{\text{omit}} = \mathbb{E}\bigl[\|X - \mathbb{E}[X \mid Z_{\text{vae}}, c]\|_2^2\bigr]. \tag{31}$$

Proof. The identity for MSE_{rec} follows from Equation 25. Since $\Sigma_{\text{omit}} \succeq 0$, we have $\Sigma_{\text{omit}} \preceq \lambda_{\max}(\Sigma_{\text{omit}}) \mathbf{I} \preceq (\operatorname{tr} \Sigma_{\text{omit}}) \mathbf{I}$, hence $B_c \Sigma_{\text{omit}} B_c^\top \preceq \text{MSE}_{\text{rec}}\, B_c B_c^\top \preceq \text{MSE}_{\text{rec}} \|B_c\|_2^2 \mathbf{I}$. Conjugating by $\Sigma_\xi^{-1/2}$ and using $\Sigma_\xi^{-1} \preceq \lambda_{\min}(\Sigma_\xi)^{-1} \mathbf{I}$ gives

$$\Sigma_\xi^{-1/2} B_c \Sigma_{\text{omit}} B_c^\top \Sigma_\xi^{-1/2} \preceq \frac{\|B_c\|_2^2 \,\text{MSE}_{\text{rec}}}{\lambda_{\min}(\Sigma_\xi)} \mathbf{I}. \tag{32}$$

The log determinant is monotone in the positive semidefinite order, so substituting this bound into Equation 21 and evaluating the determinant of the resulting multiple of the identity gives the claim.

Because $\mathrm{MSE}_{\mathrm{rec}}$ is the minimum mean squared error of predicting X from Z_{vae}, it is bounded above by the error of any particular decoder, including one acting on the sampled latent $\widetilde{Z}$, since $X \to Z_{\mathrm{vae}} \to \widetilde{Z}$ forms a Markov chain. Measured reconstruction fidelity therefore upper bounds the information loss term within this model. Corollary 1 operates in the regime where the action sensitivity $\|B_c\|_2^2/\lambda_{\min}(\Sigma_\xi)$ diverges and the bound becomes vacuous, so the two results are compatible. The bound relates reconstruction error to action information loss through the action sensitivity and noise level in this linear Gaussian model.

A.4 Limits of Downstream Recovery

Let a downstream model construct a hidden state R deterministically from (Z, c). The data processing inequality gives

$$I(S; R) \le I(S; Z, c). \tag{33}$$

Thus, a trainable transformer can reorganize retained information into a predictor state, but it cannot recreate action-relevant information discarded jointly by the representation and context.

B DemoDPO Derivation and Algorithm

We derive the DemoDPO objective by following Diffusion-DPO from action likelihoods to a surrogate based on denoising errors, then converting these errors to flow matching errors. The derivation separates the path approximation, the bound used for sampling a single step, and the weighting choice used in training.

B.1 Objective Derivation

1. Preference objective. Let $\mathcal{D} = \{(x_t, A^w, A^l)\}$ contain preference pairs under the training condition $x_t = (X_t, c_t)$. Using the notation of Equation 8, standard DPO optimizes

$$\mathcal{L}_{\mathrm{DPO}} = -\mathbb{E}_{\mathcal{D}} \log \sigma \left(\beta \left[\rho_\theta(A^w \mid x_t) - \rho_\theta(A^l \mid x_t) \right] \right), \qquad \rho_\theta(A \mid x_t) = \log \frac{\pi_\theta(A \mid x_t)}{\pi_{\mathrm{ref}}(A \mid x_t)}. \tag{34}$$

This objective favors the preferred action relative to the reference policy. Its direct evaluation requires action likelihoods, which are not outputs of the velocity predictor. We therefore follow the diffusion surrogate construction of Wallace et al. (2024), before adapting it to flow matching.

2. From actions to generation paths. In the diffusion construction, let $A_0 = A$ and let $A_{1:K}$ denote intermediate noisy actions. Here k indexes diffusion steps, not control steps. For $\phi \in \{\theta, \mathrm{ref}\}$, the path distribution and its endpoint marginal are

$$p_\phi(A_{0:K} \mid x_t) = p(A_K) \prod_{k=1}^{K} p_\phi(A_{k-1} \mid A_k, x_t), \qquad p_\phi(A \mid x_t) = \int p_\phi(A_{0:K} \mid x_t)\, dA_{1:K}. \tag{35}$$

The marginal integrates over all paths ending at A, whereas the path distribution factorizes into individual transitions. Diffusion-DPO constructs a preference objective on paths, using the joint KL to upper bound the endpoint KL:

$$D_{\mathrm{KL}}\left(p_\theta(A_0 \mid x_t) \| p_{\mathrm{ref}}(A_0 \mid x_t)\right) \le D_{\mathrm{KL}}\left(p_\theta(A_{0:K} \mid x_t) \| p_{\mathrm{ref}}(A_{0:K} \mid x_t)\right). \tag{36}$$

The corresponding score averages the path log ratio conditional on the endpoint:

$$\rho_\theta^{\mathrm{path}}(A \mid x_t) = \mathbb{E}_{p_\theta(A_{1:K} \mid A, x_t)} \left[\log \frac{p_\theta(A_{0:K} \mid x_t)}{p_{\mathrm{ref}}(A_{0:K} \mid x_t)} \right]. \tag{37}$$

Replacing ρ_θ by this score gives a surrogate preference objective on paths. The bound in Equation 36 applies to the endpoint KL, not the original DPO loss. The shared terminal prior cancels in the path ratio, leaving a sum of transition log ratios.

3. From paths to a sampled step. Sampling paths conditional on a given endpoint remains intractable. Following Diffusion-DPO, we approximate this posterior by the known forward noising process $q(A_{1:K} \mid A)$. Define the expected transition log ratio

$$g_{\theta,k}(A, A_k \mid x_t) := \mathbb{E}_{q(A_{k-1}|A_k,A)}\left[\log \frac{p_\theta(A_{k-1} \mid A_k, x_t)}{p_{\text{ref}}(A_{k-1} \mid A_k, x_t)}\right]. \tag{38}$$

At $k = 1$, $A_0 = A$ is fixed, so this is the log density ratio for the final denoising step. The approximated path score becomes

$$\rho^q_\theta(A \mid x_t) := \mathbb{E}_{q(A_{1:K}|A)} \log \frac{p_\theta(A_{0:K} \mid x_t)}{p_{\text{ref}}(A_{0:K} \mid x_t)} = K\, \mathbb{E}_{k,A_k\sim q(\cdot|A)} g_{\theta,k}(A, A_k \mid x_t), \tag{39}$$

where k is uniform on $\{1, \dots, K\}$. Thus, the path sum can be estimated by sampling a single step. For a fixed pair, write $g^w_{\theta,k}$ and $g^l_{\theta,k}$ for the two transition scores at the same sampled step. Since $-\log\sigma$ is convex, Jensen's inequality gives

$$\mathcal{L}_q := -\mathbb{E}_{\mathcal{D}} \log\sigma\left(\beta K\, \mathbb{E}_{k,q}[g^w_{\theta,k} - g^l_{\theta,k}]\right) \le -\mathbb{E}_{\mathcal{D},k,q} \log\sigma\left(\beta K[g^w_{\theta,k} - g^l_{\theta,k}]\right) =: \mathcal{L}_{\text{step}}. \tag{40}$$

Here q supplies the noisy marginals for both actions. This bound makes training possible with one sampled noise level per pair. It applies to the objective after the forward process approximation.

4. From Gaussian transitions to denoising errors. The remaining transition scores can be evaluated through noise prediction. For $k \ge 2$, inserting the forward posterior into the log ratio gives

$$\begin{aligned} g_{\theta,k}(A, A_k \mid x_t) = {} & D_{\text{KL}}\left(q(A_{k-1} \mid A_k, A) \| p_{\text{ref}}(A_{k-1} \mid A_k, x_t)\right) \\ & - D_{\text{KL}}\left(q(A_{k-1} \mid A_k, A) \| p_\theta(A_{k-1} \mid A_k, x_t)\right). \end{aligned} \tag{41}$$

In the Gaussian diffusion construction, the forward posterior has mean $\mu_{q,k}$, and the reverse models have means $\mu_{\phi,k}$ and a shared fixed covariance $\nu_k^2\mathbf{I}$. Terms involving the posterior covariance cancel between the two KLs. Under the usual noise parameterization, $\mu_{\phi,k} - \mu_{q,k} = b_k(\epsilon - \hat\epsilon_\phi)$, where b_k depends only on the noise schedule. Consequently,

$$\begin{aligned} g_{\theta,k}(A, A_k \mid x_t) &= -\frac{1}{2\nu_k^2}\left[\|\mu_{\theta,k} - \mu_{q,k}\|_2^2 - \|\mu_{\text{ref},k} - \mu_{q,k}\|_2^2\right] \\ &= -\omega_k\left[\|\hat\epsilon_\theta - \epsilon\|_2^2 - \|\hat\epsilon_{\text{ref}} - \epsilon\|_2^2\right], \qquad \omega_k = \frac{b_k^2}{2\nu_k^2}. \end{aligned} \tag{42}$$

The noise predictors are evaluated on the same A_k, condition x_t, and step k. For a Gaussian endpoint decoder with shared fixed variance, the $k = 1$ log density ratio likewise reduces to a difference of reconstruction errors, which can be expressed as noise errors with its corresponding weight. Substituting these scores into Equation 40 yields the denoising error surrogate: the preferred action should have a smaller error relative to the reference model than the rejected action.

5. From denoising to flow matching. To adapt this surrogate to our velocity predictor, we follow Liu et al. (2026) and use the linear interpolation from Equation 5:

$$A'_\tau = (1-\tau)A + \tau\epsilon, \qquad v_\tau = \epsilon - A, \qquad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{43}$$

Since $\epsilon = A'_\tau + (1-\tau)v_\tau$, the velocity predictor induces a noise predictor through

$$\hat\epsilon_\phi := A'_\tau + (1-\tau)\hat v_\phi(A'_\tau, x_t, \tau), \qquad \phi \in \{\theta, \text{ref}\}. \tag{44}$$

Subtracting the true noise gives the exact error conversion

$$\|\hat\epsilon_\phi - \epsilon\|_2^2 = (1-\tau)^2\|\hat v_\phi(A'_\tau, x_t, \tau) - v_\tau\|_2^2. \tag{45}$$

Thus, the denoising surrogate becomes a preference loss on velocity errors with a weight that depends on τ. Following the constant weighting used by Flow-DPO, we replace the resulting weight by a constant. This is a choice of training objective, separate from the exact error conversion above.

6. DemoDPO objective. We now use the normalization and score from Equations 9 and 10:

$$\bar\ell_\phi(A \mid x_t; \tau, \epsilon) := \frac{1}{H_a D_a}\|\hat v_\phi(A'_\tau, x_t, \tau) - (\epsilon - A)\|_2^2, \tag{46}$$

$$s_\theta(A \mid x_t; \tau, \epsilon) := -\frac{1}{2}\left[\bar\ell_\theta(A \mid x_t; \tau, \epsilon) - \bar\ell_{\text{ref}}(A \mid x_t; \tau, \epsilon)\right]. \tag{47}$$

All constant scale factors are absorbed into β, with $1/2$ kept explicit to match the main text. For each candidate pair, we sample $\tau \sim \text{Unif}[0,1]$ and a shared $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, independently of the

noises used to generate the candidates. We form $A_\tau^{\prime,s} = (1-\tau)A^s + \tau\epsilon$ for $s \in \{w, l\}$. Using demonstration MSE to rank candidates and retaining pairs with $M_{\text{pair}} = 1$ gives

$$\mathcal{L}_{\text{DemoDPO}} = -\mathbb{E}_{\mathcal{D},\tau,\epsilon}\left[\log\sigma\left(\beta\left[s_\theta(A^w \mid x_t;\tau,\epsilon) - s_\theta(A^l \mid x_t;\tau,\epsilon)\right]\right) \mid M_{\text{pair}} = 1\right]. \quad (48)$$

This is Equation 11: the sampled preference loss uses velocity errors, a frozen reference policy, and demonstration rankings, without evaluating action likelihoods or complete generation paths.

B.2 Training Algorithm

Algorithm 1 summarizes one DemoDPO update. Sampling and candidate selection are performed independently for each training example. `mse` averages over valid action entries while retaining the batch dimension and, for candidate ranking, the candidate dimension. `M` denotes M_{pair}. `sample_time()` draws $\tau \sim \text{Unif}[0, 1]$ independently for each example, with the same τ and ϵ shared by both candidates.

Algorithm 1 DemoDPO with Demonstration-Guided Group Ranking

```
# fn, ref: velocity predictor and frozen reference
# x, A_demo: condition and demonstrated actions
# G, pair_epsilon, beta: candidate count, pair threshold, DPO coefficient

A = stopgrad(sample(ref, x, G))          # G candidates per condition
d = mse(A, A_demo)                       # demonstration error
w, l = argmin(d), argmax(d)
M = d[l] - d[w] >= pair_epsilon          # retain distinct pairs
A_w, A_l = A[w], A[l]

tau = sample_time()
eps = randn_like(A_demo)                 # shared by both candidates
z_w = (1 - tau) * A_w + tau * eps
z_l = (1 - tau) * A_l + tau * eps
v_w, v_l = eps - A_w, eps - A_l

err_w = mse(fn(z_w, x, tau), v_w)
err_l = mse(fn(z_l, x, tau), v_l)
ref_w = mse(ref(z_w, x, tau), v_w)
ref_l = mse(ref(z_l, x, tau), v_l)

s_w = -0.5 * (err_w - stopgrad(ref_w))   # relative preference score
s_l = -0.5 * (err_l - stopgrad(ref_l))
loss = -logsigmoid(beta * (s_w - s_l))
loss = (loss * M).sum() / M.sum().clamp(min=1)
```

C Implementation Details

We provide the training sample construction, frozen encoder comparison, evaluation protocols, visual representation analysis, and per task RoboTwin results.

C.1 Training Sample Construction

Following Fast-WAM (Yuan et al., 2026), we concatenate images from the head and two wrist cameras into a single image before feeding it to the vision encoder. Because our study aims to isolate the effect of visual representations, we instantiate the task context c_t as a discrete task ID and use the same conditioning across all controlled comparisons, rather than natural language instructions. At control step t, each training sample contains the current visual observation X_t and an action chunk $A_t = (a_t, \ldots, a_{t+H_a-1})$, where $a_t \in \mathbb{R}^{D_a}$ and D_a is the action dimension. During training, we randomly sample one future offset $\delta \in \{1, \ldots, H_a\}$ and use the corresponding observation $X_{t+\delta}$ for future embedding prediction. Thus, we use $N = 1$ future target per training sample.

Sampling offsets across training provides supervision over the prediction horizon while limiting the number of future query tokens. Empirically, increasing N provides limited performance gains while introducing more future query tokens and slowing training. All LeWAM variants are trained only on the RoboTwin demonstrations described in Section 4.1, while retaining the original visual pretraining of the frozen I-JEPA encoder.

C.2 Frozen Encoder Comparison

The goal of this comparison is to select a frozen visual encoder for LeWAM under a fixed downstream action predictor. We compare representative checkpoints from each encoder family, with different architectures and parameter counts. We first establish two representative reference points: the Wan2.2 VAE, which provides the compressed latent used by WAMs built on VDMs, and SigLIP (Zhai et al., 2023), a widely adopted visual encoder in VLAs. We then compare MAE-Large (He et al., 2022), DINOv3-Large, and V-JEPA2-Large (Assran et al., 2025). Based on this comparison, we further evaluate V-JEPA2-Huge and I-JEPA-Huge (Assran et al., 2023).

For each encoder, we freeze all pretrained parameters and independently train the same lightweight DiT-style action predictor on RoboTwin. Future embedding prediction is disabled so that the experiment evaluates each representation solely through action generation. The action predictor architecture, current observations from multiple views, task conditioning, action horizon, optimizer, batch size, and training schedule are kept fixed across encoders. Only the encoder-specific input projection and the downstream action predictor are trained. All encoder variants use a 12-layer action predictor with a hidden size of 1280, and each encoder output is mapped to this width using a learned linear projection. After selecting I-JEPA-H/14, we use its frozen 0.6B-parameter encoder in LeWAM with an input resolution of 224×224 and a patch size of 14×14, producing 256 image tokens. The predictor and AdaFuse contain 402M trainable parameters, giving LeWAM 1.0B parameters in total. Following DiT (Peebles & Xie, 2023), we use AdaLN-Zero to condition the predictor on noise timestep and task embeddings. Noisy action tokens use the sampled noise timestep, while visual tokens, clean action tokens, and future queries use the time embedding evaluated at zero. Before fusion, a shared LayerNorm independently normalizes each encoder layer's tokens across channels.

Each observation is processed with the input range, spatial resolution, and normalization for each channel expected by its pretrained encoder. For consistency with the other encoders, we use only the current image, repeated 16 times following V-JEPA2's official single-image protocol (Assran et al., 2025). The complete action generation and probing results are reported in Table 1 and discussed in RQ1.

C.3 DemoDPO Ablation Evaluation Protocol

The DemoDPO hyperparameter ablations in Figure 5 use a fixed evaluation subset of eight challenging RoboTwin tasks to reduce evaluation cost. All policies are trained on the complete RoboTwin 2.0 training set covering 50 tasks, comprising 2,500 clean and 25,000 randomized demonstrations, as described in Section 4.1. The subset is used only for evaluation and is not a training split by task. For each task, we evaluate 100 clean and 100 randomized cases, resulting in 1,600 evaluation cases in total. In alphabetical order, the task list is `blocks_ranking_size`, `hanging_mug`, `open_microwave`, `pick_diverse_bottles`, `place_a2b_right`, `place_can_basket`, `stamp_seal`, and `turn_switch`.

Under this evaluation protocol, the sweeps in Figure 5, from *(a)* to *(c)*, vary $\varepsilon_{\text{pair}}$ with $\beta = 2000$ and $G = 4$, β with $\varepsilon_{\text{pair}} = 10^{-3}$ and $G = 4$, and G with $\varepsilon_{\text{pair}} = 10^{-3}$ and $\beta = 5000$, respectively.

C.4 Inference Efficiency

We benchmark all models on an NVIDIA H800 with batch size 1 and 10 denoising steps. Latency includes visual encoding, language encoding and KV-cache prefilling where applicable, and complete action denoising, with images re-encoded for every request. CPU image preprocessing, tokenization, CPU–GPU transfers, robot execution, and one-time CUDA Graph capture are excluded. We report peak PyTorch-allocated GPU memory with CUDA Graph enabled, including model parameters and graph-held tensors but excluding capture-phase peaks.

Table 3 shows that LeWAM achieves the lowest latency with CUDA Graph disabled or enabled and the lowest peak memory among the evaluated models.

Table 3: GPU inference efficiency. Peak allocated memory is measured with CUDA Graph enabled.

Model	Latency (ms) ↓		Peak memory	Action chunk
	CUDA Graph off	CUDA Graph on	(GiB) ↓	length
$\pi_{0.5}$ (PyTorch)	162.45	53.14	7.037	50
LingBot-VLA-v2	711.31	126.53	12.089	50
Fast-WAM	325.55	102.38	23.198	32
LeWAM	**103.53**	**31.90**	**1.988**	32

C.5 Visual Representation Analysis

Policy attention visualization. Attention scores are averaged across the action tokens, all attention heads, and all transformer layers at each inference step. Figure 6 presents attention maps across several manipulation tasks using the same aggregation protocol as Figure 4. In Blocks Ranking RGB and Blocks Ranking Size, attention shifts between blocks as the policy grasps and places them in sequence. Prominent responses appear around the active block, gripper, and placement region in both clean and cluttered scenes. These observations suggest that action generation emphasizes object locations and spatial relationships, motivating the depth and segmentation probes below.

Prior work probes pretrained visual encoders (Zhu et al., 2023; Chen et al., 2024; Assran et al., 2023; Mur-Labadia et al., 2026). We analyze the information accessible from each frozen visual representation using depth and segmentation probes. We randomly sample one timestep from each RoboTwin 2.0 training episode, obtaining 27,500 samples with synchronized front, left, and right views. The samples are divided into training, validation, and test sets using an $8{:}1{:}1$ ratio. All visual encoders remain frozen. For AdaFuse, probing uses frozen fusion weights learned by LeWAM, while action success in Table 1 uses action-only training. Each probe applies LayerNorm, a linear projection to a width of 1280, and GELU to the frozen visual tokens. The depth and segmentation mask heads operate on the spatial tokens, while the mask confidence head applies mean pooling before linear prediction. Only the projection and probe heads are trained.

For depth probing, DA3MONO-LARGE (Lin et al., 2025) generates a relative depth target independently for each view. The targets are standardized within each view and arranged in the same spatial layout as the visual input. The depth probe is trained with mean squared error over valid pixels. We evaluate depth prediction using scale and shift invariant root mean squared error (SSI-RMSE, shown as RMSE in Table 1) and Spearman rank correlation (Corr.) between the predicted and teacher depth values.

SAM3 (Carion et al., 2026) generates instance masks and confidence targets independently for each view. Predicted masks are assigned to the teacher masks using Hungarian matching. The training objective combines binary cross entropy (BCE) for mask and confidence prediction with Dice loss. We report average precision at intersection over union (IoU) thresholds of 0.50 (AP@50) and 0.75 (AP@75). A predicted mask is correct when its IoU with a teacher mask exceeds the threshold.

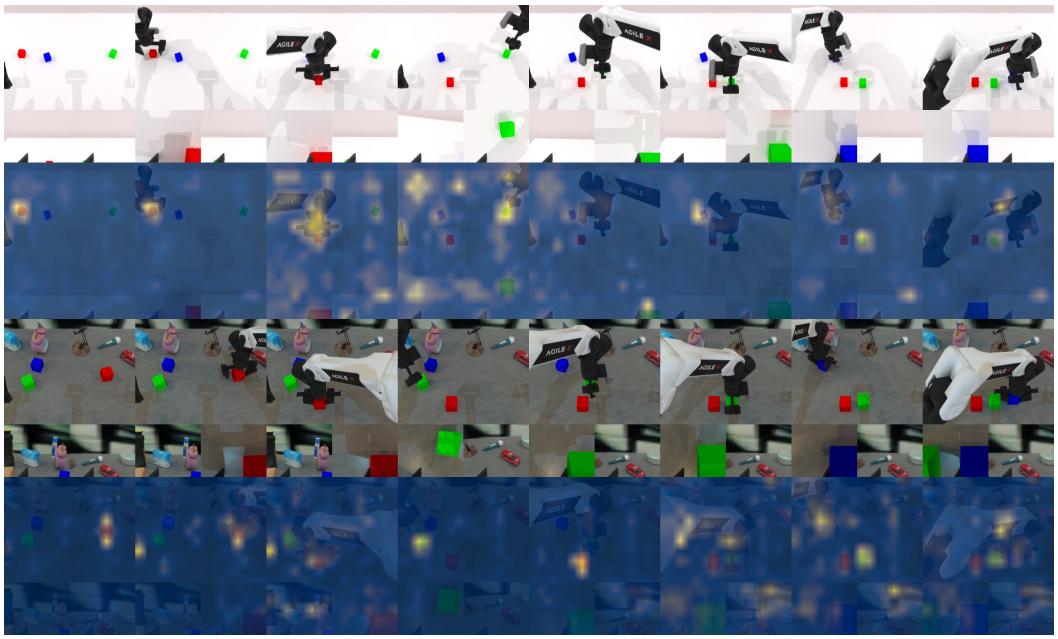
Table 1 reports the probing results together with action success. Among the individual encoders, I-JEPA-Huge performs best on both depth and segmentation. The AdaFuse representation further improves all depth and segmentation metrics relative to the final I-JEPA layer. Figure 7 presents representative depth and segmentation predictions.

C.6 RoboTwin Per-Task Results

Table 4 reports success rates on all 50 RoboTwin 2.0 tasks under clean and randomized settings. In Table 4, results for GigaWorld-Policy and LaWAM are taken from their original papers (Ye et al., 2026a; Chen et al., 2026), while results for all other external baselines are taken from Yuan et al. (2026). LeWAM achieves an overall success rate of 92.28%, with 93.14% in clean settings and 91.42% under randomization, matching the performance of leading VLA and WAM baselines.

Blocks Ranking RGB

Place the red block, green block, and blue block in the order of red, green, and blue from left to right, placing in a row.



Blocks Ranking Size

There are three blocks on the table, the color of the blocks is random, move the blocks to the center of the table, and arrange them from largest to smallest, from left to right.

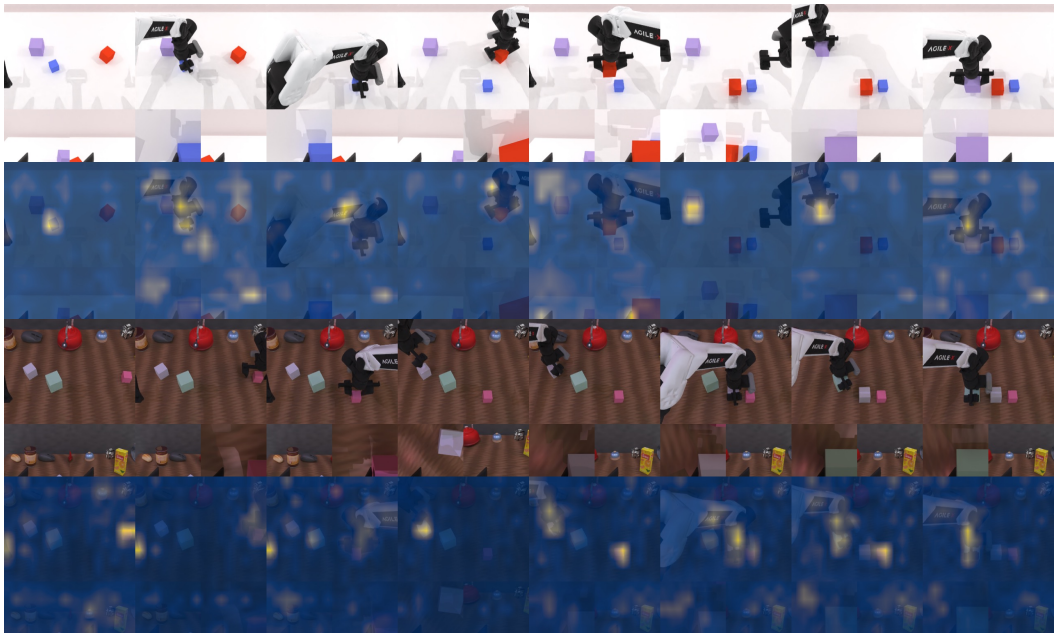


Figure 6: Policy attention visualizations under clean and randomized settings. Attention follows the active objects and interaction regions as the manipulation progresses.

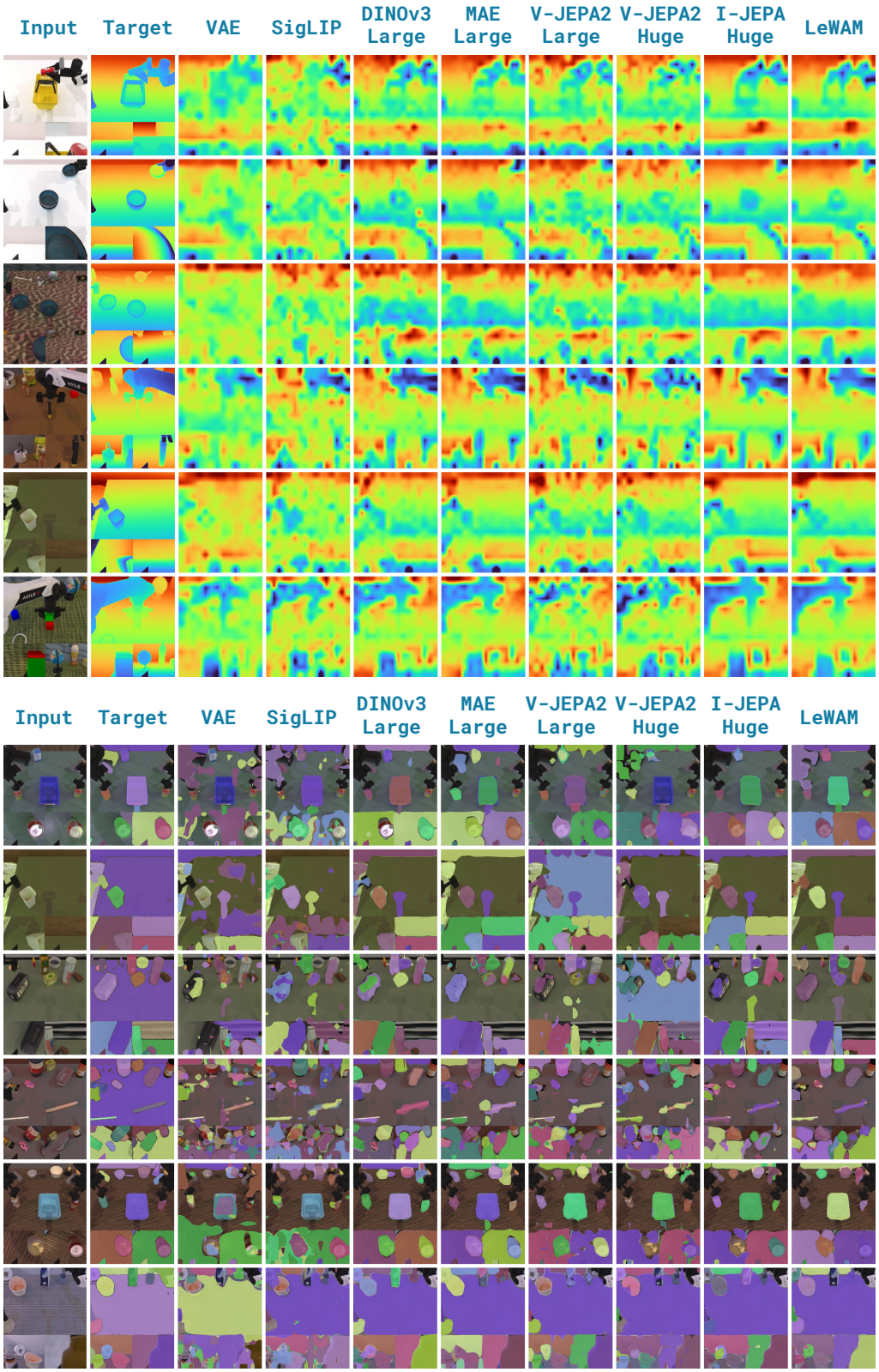


Figure 7: Depth and segmentation probing across visual representations. LeWAM denotes I-JEPA with learned AdaFuse weights. Depth maps share a fixed color scale, while segmentation colors distinguish local mask instances rather than semantic categories.

Table 4: Per-task success rates (%) on the 50 RoboTwin 2.0 tasks under clean and randomized evaluation settings. Best results are shown in bold.

Task	$\pi_{0.5}$		GigaWorld-Policy		Motus		LaWAM		Fast-WAM		LingBot-VA		LeWAM	
	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
Adjust Bottle	**100**	99	**100**	**100**	89	93	**100**	**100**	**100**	**100**	90	94	**100**	**100**
Beat Block Hammer	96	93	86	86	95	88	90	93	**99**	97	96	**98**	**99**	97
Blocks Ranking RGB	92	85	92	96	99	97	97	**100**	**100**	**100**	99	98	**100**	99
Blocks Ranking Size	49	26	44	48	75	63	93	89	**94**	**98**	**94**	96	81	80
Click Alarmclock	98	89	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	99	**100**	**100**	98
Click Bell	99	66	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Dump Bin Bigbin	92	97	92	**100**	95	91	**97**	95	**97**	96	89	96	**97**	98
Grab Roller	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Handover Block	66	57	80	80	86	73	96	87	95	81	99	78	**100**	**93**
Handover Mic	98	97	72	72	78	63	93	98	**99**	**100**	94	96	**99**	99
Hanging Mug	18	17	16	12	38	38	51	43	58	**62**	40	28	**65**	53
Lift Pot	96	85	98	98	96	99	**100**	99	**100**	**100**	**100**	99	**100**	**100**
Move Can Pot	51	55	76	78	34	74	**98**	93	90	88	94	**97**	93	93
Move Pillbottle Pad	84	61	90	90	93	96	97	90	**100**	**99**	99	**99**	97	96
Move Playingcard Away	96	84	78	72	**100**	96	**100**	**100**	**100**	**100**	**100**	99	**100**	**100**
Move Stapler Pad	56	42	92	82	83	85	**94**	**87**	77	64	91	79	93	85
Open Laptop	90	96	96	98	95	91	**100**	**100**	98	**100**	92	94	99	**100**
Open Microwave	34	77	74	66	**95**	**91**	41	43	62	45	82	86	87	86
Pick Diverse Bottles	81	71	82	70	90	**91**	**91**	88	80	85	89	82	87	81
Pick Dual Bottles	93	63	86	86	96	90	**100**	95	**100**	96	**100**	**99**	99	94
Place A2B Left	87	82	94	88	88	79	**98**	91	95	**93**	97	**93**	81	76
Place A2B Right	87	84	90	92	91	87	89	94	93	**99**	**97**	95	85	80
Place Bread Basket	77	64	82	82	91	94	92	85	91	93	**97**	**95**	**97**	**95**
Place Bread Skillet	85	66	94	90	86	83	90	83	90	**93**	**95**	90	94	85
Place Burger Fries	94	87	98	96	98	98	93	96	96	**99**	97	95	**99**	98
Place Can Basket	62	62	78	74	81	76	**92**	65	71	69	81	**84**	82	67
Place Cans Plasticbox	94	84	**100**	**100**	98	94	**100**	95	99	96	**100**	99	**100**	99
Place Container Plate	99	95	98	96	98	99	**100**	**100**	96	**100**	99	97	**100**	**100**
Place Dual Shoes	75	75	96	84	93	87	**98**	**94**	94	88	94	89	86	90
Place Empty Cup	**100**	99	90	90	99	98	99	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Place Fan	87	85	92	94	91	87	92	93	96	**96**	**99**	93	96	94
Place Mouse Pad	60	39	88	90	66	68	91	84	83	89	**93**	**96**	**93**	85
Place Object Basket	80	76	90	**92**	81	87	**92**	90	89	88	91	88	89	90
Place Object Scale	86	80	88	80	88	85	95	88	90	**97**	**96**	95	89	95
Place Object Stand	91	85	**100**	**98**	98	97	92	93	90	94	99	96	98	96
Place Phone Stand	81	81	82	72	87	86	93	94	**97**	**99**	**97**	97	**97**	98
Place Shoe	92	93	98	96	99	97	**100**	**100**	96	99	98	98	99	97
Press Stapler	87	83	96	96	93	**98**	**98**	97	90	97	85	82	83	85
Put Bottles Dustbin	84	79	72	70	81	79	94	**92**	**95**	90	87	91	94	90
Put Object Cabinet	80	79	74	74	88	71	90	82	**94**	**89**	85	87	91	83
Rotate QRcode	89	87	90	84	89	73	94	89	93	89	**96**	91	93	**92**
Scan Object	72	65	60	64	67	66	**96**	90	89	**92**	**96**	91	89	90
Shake Bottle	99	97	**100**	98	**100**	97	**100**	**100**	**100**	**100**	**100**	97	**100**	99
Shake Bottle Horizontally	99	99	**100**	**100**	**100**	98	**100**	**100**	**100**	**100**	**100**	99	**100**	98
Stack Blocks Three	91	76	70	78	91	95	90	75	95	97	99	**98**	**100**	96
Stack Blocks Two	97	**100**	**100**	94	**100**	98	**100**	97	**100**	**100**	**100**	98	**100**	**100**
Stack Bowls Three	77	71	70	72	79	**87**	**90**	80	80	81	86	83	85	86
Stack Bowls Two	95	96	96	92	98	98	**100**	**99**	92	98	94	98	99	95
Stamp Seal	79	55	**96**	**98**	93	92	89	88	90	94	**96**	97	82	92
Turn Switch	62	54	82	**84**	**84**	78	47	56	61	59	44	45	60	68
Average	82.74	76.76	86.36	85.04	88.66	87.02	92.64	89.80	91.88	**91.78**	92.90	91.50	**93.14**	91.42
Overall	79.75		85.70		87.84		91.22		91.83		92.20		**92.28**	

C.7 LeWAM Per-Task Ablations

Table 5 reports per-task results for the LeWAM ablations. Direct future embedding prediction provides a modest overall gain over the Action Only baseline, improving the success rate from 86.17% to 87.02%, while the flow matching variant reaches 86.21%. Adding AdaFuse produces a larger improvement to 90.69%. DemoDPO further raises the success rate to 92.28%. Compared with the AdaFuse variant, it improves 25 clean and 24 randomized per-task results. Relative to the Action Only baseline, the complete LeWAM improves 36 clean and 41 randomized per-task scores, showing that the gain is distributed across many tasks rather than concentrated in a few cases.

Table 5: Per-task LeWAM ablations on RoboTwin 2.0. LeWAM uses I-JEPA-H/14 with 402M trainable parameters. WM (FM) uses flow matching for future embedding prediction. The remaining variants progressively add WM, AdaFuse, and DemoDPO. Best results are shown in bold.

Task	Action Only		+ WM (FM)		+ WM		+ WM + AdaFuse		LeWAM	
	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
Adjust Bottle	99	**100**	**100**	99	**100**	99	**100**	**100**	**100**	**100**
Beat Block Hammer	98	92	98	94	96	**97**	**99**	94	**99**	**97**
Blocks Ranking RGB	98	96	99	98	99	98	**100**	**99**	**100**	**99**
Blocks Ranking Size	78	63	77	76	77	68	79	**80**	**81**	**80**
Click Alarmclock	**100**	96	**100**	98	**100**	**99**	**100**	98	**100**	98
Click Bell	**100**	99	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Dump Bin Bigbin	95	95	**100**	97	94	97	97	94	97	**98**
Grab Roller	99	99	99	**100**	**100**	99	**100**	**100**	**100**	**100**
Handover Block	96	82	78	78	93	72	96	81	**100**	**93**
Handover Mic	**99**	98	98	98	**99**	94	98	**99**	**99**	**99**
Hanging Mug	**65**	46	24	12	47	38	46	**54**	**65**	53
Lift Pot	**100**	69	**100**	68	99	72	**100**	**100**	**100**	**100**
Move Can Pot	79	75	88	86	**98**	**94**	89	92	93	93
Move Pillbottle Pad	84	89	94	92	96	94	**97**	**96**	**97**	**96**
Move Playingcard Away	99	99	**100**	98	**100**	**100**	**100**	99	**100**	**100**
Move Stapler Pad	71	62	71	70	75	80	**94**	**86**	93	85
Open Laptop	96	**100**	98	98	94	**100**	98	**100**	**99**	**100**
Open Microwave	76	69	**89**	**91**	67	67	76	82	87	86
Pick Diverse Bottles	72	75	66	68	73	66	83	78	**87**	**81**
Pick Dual Bottles	98	91	94	90	91	92	**99**	**94**	**99**	**94**
Place A2B Left	82	70	81	73	**84**	**76**	79	71	81	**76**
Place A2B Right	76	63	81	65	79	67	83	79	**85**	**80**
Place Bread Basket	91	90	90	86	88	93	92	92	**97**	**95**
Place Bread Skillet	94	79	91	81	91	86	**97**	**87**	94	85
Place Burger Fries	97	98	98	93	96	95	**99**	**100**	**99**	98
Place Can Basket	46	53	73	61	73	61	75	62	**82**	**67**
Place Cans Plasticbox	99	95	**100**	**99**	99	93	**100**	97	**100**	**99**
Place Container Plate	99	94	97	94	99	93	**100**	**100**	**100**	**100**
Place Dual Shoes	83	80	**88**	86	76	80	81	84	86	**90**
Place Empty Cup	**100**	**100**	**100**	98	**100**	99	**100**	**100**	**100**	**100**
Place Fan	87	86	**97**	**94**	96	89	**97**	90	96	**94**
Place Mouse Pad	76	75	77	74	87	**85**	89	84	**93**	**85**
Place Object Basket	**94**	83	79	77	80	74	90	**90**	89	**90**
Place Object Scale	83	75	86	80	82	80	85	92	**89**	**95**
Place Object Stand	85	93	92	90	92	90	94	95	**98**	**96**
Place Phone Stand	96	84	94	81	**97**	82	**97**	**98**	**97**	**98**
Place Shoe	93	93	93	92	94	94	**99**	94	**99**	**97**
Press Stapler	**83**	79	80	75	80	83	79	**85**	**83**	**85**
Put Bottles Dustbin	88	74	92	76	87	73	**94**	89	**94**	**90**
Put Object Cabinet	**92**	**84**	81	81	81	80	90	77	91	83
Rotate QRcode	94	90	**96**	**93**	84	90	93	87	93	92
Scan Object	88	73	87	75	88	72	87	87	**89**	**90**
Shake Bottle	99	**100**	99	**100**	99	**100**	99	98	**100**	99
Shake Bottle Horizontally	**100**	**100**	**100**	**100**	99	**100**	99	99	**100**	98
Stack Blocks Three	96	95	99	91	**100**	**99**	99	97	**100**	96
Stack Blocks Two	99	99	**100**	**100**	**100**	98	**100**	**100**	**100**	**100**
Stack Bowls Three	83	78	78	79	81	70	78	**89**	**85**	86
Stack Bowls Two	97	95	97	90	98	95	98	**97**	**99**	95
Stamp Seal	53	53	53	48	77	66	79	86	**82**	**92**
Turn Switch	**66**	70	57	69	58	70	63	**71**	60	68
Average	88.42	83.92	88.18	84.24	88.86	85.18	91.32	90.06	**93.14**	**91.42**
Overall	86.17		86.21		87.02		90.69		**92.28**	

D REAL-WORLD EXPERIMENTS

Data collection. We conduct the real-world experiments on an AgileX dual arm platform consisting of two Piper manipulators. Demonstrations are collected through leader-follower teleoperation. The policy observes synchronized RGB images from a fixed front camera and cameras mounted on the left and right robot arms. We collect 400 episodes for each of Stack Blocks, Fold Towel, and Arrange Flowers, resulting in 1,200 episodes and approximately 11 hours of data collected in house. Both observations and actions are recorded at 30 Hz. The initial object positions are varied across episodes. All demonstrations use delta joint positions as the action representation.

Policy training. All evaluated methods are trained on the same demonstrations for 20,000 optimization steps. The baselines are finetuned from their publicly available pretrained checkpoints. We use AdamW with a learning rate of 5×10^{-5} and a batch size of 256 for all methods. Each policy predicts an action chunk of 50 steps. We use the checkpoints after 20,000 supervised optimization steps for baseline evaluation. For LeWAM, we further apply DemoDPO for 1,000 updates using the same hyperparameters as in Section 4.1, including $G = 4$ and $\beta = 5000$, and evaluate the refined policy.

Robot evaluation. All policies are deployed on an NVIDIA A100 GPU with 40 GB of memory and executed at 30 Hz. Before each rollout, both robot arms are returned to the same home configuration. The objects are then placed in the initial configuration sampled for that rollout. Each method is evaluated for 50 rollouts on each of the three tasks. Each rollout is capped at three minutes.

Task progress. The three real-world tasks contain several sequential manipulation stages. We measure task progress by scoring each recorded rollout according to completed milestones.

Let $s_{m,q,i}$ denote the score of rollout i produced by method m on task q, and let M_q denote the maximum score for that task. We report task progress as

$$\text{Progress}(m, q) = \frac{100}{N_q M_q} \sum_{i=1}^{N_q} s_{m,q,i}, \tag{49}$$

where $N_q = 50$. Task progress is the average rollout score as a percentage of the maximum score.

Each milestone contributes one point and can be counted at most once in a rollout. Task dependent penalties are applied after adding the milestone points. The resulting score is clipped to the interval $[0, M_q]$. All methods are evaluated from their recorded rollouts using the same scoring rules.

Stack Blocks ($M_q = 7$). The objective is to form a three block stack with the red block at the bottom, the green block in the middle, and the blue block on top. The rollout receives one point for each of the following milestones:

1. The robot lifts the red block and moves it toward the target location.
2. The red block is placed stably.
3. The robot lifts the green block and moves it toward the red block.
4. The green block is placed stably.
5. The robot lifts the blue block and moves it toward the green block.
6. The blue block is placed stably.

The six manipulation milestones contribute at most six points. The final configuration receives one additional point when the stack is stable and neatly aligned, and no additional point when it is stable but untidy. Two points are deducted if the final structure collapses. One point is deducted if the structure collapses during execution and the robot does not continue the task.

Fold Towel ($M_q = 6$). The objective is to fold a towel and hang it on a rack. The rollout receives one point for each of the following milestones:

1. Most of the towel is unfolded.

2. The towel is fully flattened.
3. The towel is folded in half.
4. The robot adjusts the orientation of the folded towel.
5. The robot lifts the towel and moves it toward the rack.
6. The towel is placed on the rack.

No adjustment is applied when the towel remains stably and neatly placed on the rack. One point is deducted if the towel falls from the rack or its final placement is irregular.

Arrange Flowers ($M_q = 6$)**.** The objective is to insert three flowers into a vase while keeping the vase upright. For each flower, one point is awarded when the robot lifts it and moves it toward the vase, and another point is awarded when it is inserted successfully. The three flowers therefore contribute at most six points. If the vase is knocked over and the robot does not restore it, the rollout terminates and one point is deducted.

D.1 Qualitative Comparisons

DemoDPO raises task progress from 81.1%, 58.3%, and 50.7% to 84.9%, 60.7%, and 53.0% on Stack Blocks, Fold Towel, and Arrange Flowers, respectively, with example rollouts in Figure 8.

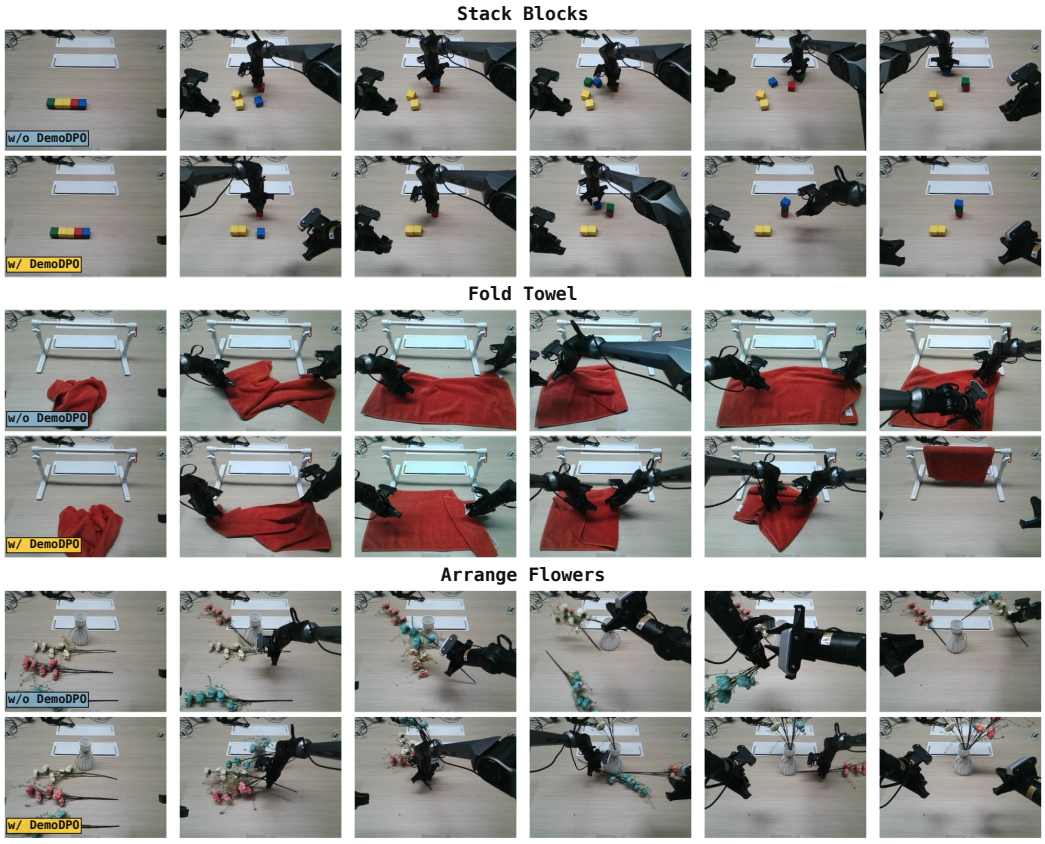


Figure 8: Qualitative effect of DemoDPO on the three real-world tasks. For each task, the two rows compare LeWAM before and after DemoDPO refinement under the same initial configuration.

Figure 9 compares representative rollouts for all four methods on the three real-world tasks. These examples visualize the partial task completion captured by the task progress metric.



Figure 9: Representative real-world rollouts for Stack Blocks, Fold Towel, and Arrange Flowers. For each task, the sequences show task execution from the fixed front camera under the same initial configuration across methods.