ARE LATENT ACTIONS TRANSFERABLE ACROSS VISUAL CONTEXTS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Reusing behavioral knowledge from unlabeled videos in new scenes requires transferable action representations. Latent action models (LAMs) learn representations from visual transitions to predict future states, but may entangle action information with source visual context. Existing benchmarks lack a common protocol for evaluating cross-context latent-action transfer. We introduce the **L**atent **A**ction **T**ransfer Benchmark (**LATBench**), a unified protocol for evaluating action transfer across visual contexts. LATBench extracts an action representation from a source video and applies it to a different target video: the source provides the action, while the target provides the visual context. It evaluates (1) *action-effect fidelity* (whether the source action effect is reproduced), (2) *target-context preservation* (whether the target object and scene are preserved), and (3) *source-content leakage* (whether source appearance is copied into the generated video). We first assess how well existing LAMs transfer across visual contexts and examine how representation design affects transferability. We then investigate the balance between action information and source content in learned representations and test the limits of transfer under longer rollouts, camera motion, and viewpoint changes. Our experiments reveal a key insight: effective transfer requires suppressing source content while retaining action information that supports state changes in the target context; lower leakage alone is insufficient.

1 INTRODUCTION

Unlabeled videos capture diverse behaviors and object interactions. They provide data for learning representations that support future-state prediction, policy learning, and robot control (Ye et al., 2025; Bu et al., 2025; Chen et al., 2026a). Reusing this experience in new scenes requires transferable action information. However, visual changes conflate actions with object appearance, backgrounds, and camera motion. Representations tied to source objects or scenes can be difficult to reuse. Learning actions from video therefore demands understanding whether the learned information transfers across visual contexts.

Latent action models (LAMs) typically employ an inverse dynamics model (IDM) to infer actions from frame pairs or visual features. They learn representations through future-state prediction and latent bottlenecks (Schmidt & Jiang, 2024; Ye et al., 2025; Yang et al., 2026). Because predicting the source video can draw on both action information and object appearance or scene structure, the learned representations may depend on the source visual context. To reuse video experience with different objects and scenes, an action representation must reproduce the action effect from a new target state while preserving the target context. We thus ask: **can latent actions learned from video reproduce action effects across visual contexts while preserving the target object and scene?**

Existing evaluations of action representations primarily take three forms: reconstruction of the source video, probing of latent features, and downstream task performance. Each, however, falls short of directly measuring whether an extracted action transfers across contexts. Reconstruction-based evaluations assess latent actions through the fidelity of future frames within the source video context (Bu et al., 2026), while LARYBench probes action information through classification and trajectory regression (Nie et al., 2026). Both approaches measure how well a representation explains or encodes an observed transition, but neither establishes that it can reproduce the action's effect in a different visual context. Downstream manipulation success depends jointly on the representa-

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.

tion, policy, and control components (Ye et al., 2025; Bu et al., 2025), making it difficult to isolate the transferability of the action representation itself. Some prior work does demonstrate cross-video transfer more directly (Chen et al., 2024; Routray et al., 2026; Garrido et al., 2026; Jiang et al., 2026; Zhang et al., 2026), but these efforts are largely qualitative; differences in generation and evaluation protocols limit direct comparisons and leave the field without a unified benchmark. Addressing this gap requires an evaluation that applies source actions to different target states and jointly assesses action effects, target preservation, and source-content leakage.

To address this evaluation gap, we introduce the **L**atent **A**ction **T**ransfer Benchmark (**LATBench**), which combines source action representations with target initial frames from different visual contexts to directly test cross-context action reuse. For comparable evaluation, we freeze representation extractors and train separate conditional decoders with a common architecture and training protocol, reducing the influence of generation components and training settings. We jointly measure action-effect fidelity (AEF), target-context preservation (TCP), and source-content leakage (SCL). These dimensions distinguish the reproduction of action effects, preservation of target content, and copying of source appearance. Using this protocol, we conduct four experimental investigations. First, we compare the transfer performance of existing LAMs. Second, we vary visual encoders, IDM inputs, and latent bottlenecks to examine how representation design affects transfer. Third, we compare feature differences and learned latent actions derived from the same encoders to disentangle the distinct roles of retaining action information and suppressing source content. Finally, we test longer rollouts, camera motion, and cross-view transfer to identify the practical limits of action reuse.

Our contributions are as follows:

- **A unified framework for cross-context action transfer.** Beyond reconstruction and action probing, LATBench directly tests action reuse in new visual contexts through joint AEF, TCP, and SCL evaluation under a common generation protocol.
- **Revealing the trade-off between retaining action information and suppressing source content.** Latent-action learning improves transfer by reducing leakage while largely preserving action fidelity. Discrete latent bottlenecks also reduce leakage but impair transfer, showing that low leakage alone does not ensure effective transfer.
- **Identifying effective design choices and transfer limitations.** V-JEPA 2.1 with CoMo-style inputs achieves the highest cross-dataset average Transfer Score. Longer rollouts degrade effective transfer, while qualitative tests expose difficulties in separating object motion from camera motion and reproducing action effects across viewpoints.

2 Related Work

Learning Latent Actions from Unlabeled Videos. Latent-action learning extracts control-oriented representations from observation sequences without action labels. LAPO learns continuous latent actions and applies vector quantization as an information bottleneck before forward dynamics prediction (Schmidt & Jiang, 2024). LAPA and Moto quantize visual transitions into discrete latent actions or motion tokens for autoregressive policy learning (Ye et al., 2025; Chen et al., 2025). IGOR learns image-goal representations designed to align actions across humans and robots (Chen et al., 2024). Continuous methods such as CoMo and CLAM retain finer transition details, but may also encode appearance, background, or camera variation (Yang et al., 2026; Liang et al., 2025). Learning Latent Action World Models in the Wild constrains continuous actions and localizes dynamics in open-domain videos, while DiLA separates dynamic structure from static content (Garrido et al., 2026; Zhang et al., 2026). Most methods predict or reconstruct source-context transitions, an objective that does not directly enforce cross-context transferability.

Latent Representations in VLAs and WAMs. Latent representations support action learning, future prediction, and reasoning for embodied control. In vision-language-action models (VLAs), LAPA and Moto use latent action pseudo-labels, UniVLA learns language-conditioned task-centric actions, villa-X jointly predicts action plans and future visual states, and ViPRA maps motion representations to robot actions (Ye et al., 2025; Chen et al., 2025; Bu et al., 2025; Chen et al., 2026b; Routray et al., 2026). LaRA-VLA internalizes textual and visual reasoning into continuous latent representations for efficient action generation (Bai et al., 2026). In world models and world-action

models (WAMs), AdaWorld adapts to new control spaces, LaWAM represents predicted dynamics as latent visual subgoals, and RepWAM models future states and actions in a shared representation space (Gao et al., 2025; Chen et al., 2026a; Wang et al., 2026). DIDO combines one-step denoising distillation with interaction-centric representation guidance to learn future dynamics representations for action prediction (Lyu et al., 2026). Performance in these systems depends on both the latent representations and their prediction or control components.

Evaluation of Latent Actions. Existing evaluations cover reconstruction, codebook statistics, latent-space structure, and downstream control. EWMBench assesses generated futures through scene consistency, motion correctness, and semantic alignment (Yue et al., 2025), whereas LARY-Bench probes action classification and trajectory regression (Nie et al., 2026). IGOR, ViPRA, Olaf-World, and DiLA evaluate transfer within their respective frameworks (Chen et al., 2024; Routray et al., 2026; Jiang et al., 2026; Zhang et al., 2026). LATBench complements these efforts with a common source-to-target protocol that measures AEF, TCP, and SCL across representation families.

3 Problem Formulation

Task setup. We study whether an action representation extracted from a source video can reproduce its effect in a different visual context. Let $V^{\mathrm{src}} = (I_t^{\mathrm{src}})_{t=0}^{T}$ and $V^{\mathrm{tgt}} = (I_t^{\mathrm{tgt}})_{t=0}^{T}$ denote the source and target videos. The source provides actions to be transferred, while the target initial frame I_0^{tgt} specifies the context for generation. The source and target videos need not depict the same action. Only the target initial frame is used for generation; subsequent target frames serve as appearance references for TCP and SCL rather than as ground-truth motion or transfer outcomes.

Source action representations. A visual encoder extracts source features $f_t^{\mathrm{src}} = \mathrm{Encoder}(I_t^{\mathrm{src}})$. For each transition $t = 0, \ldots, T-1$, we consider two representations:

$$r_t^{\mathrm{src}} = \begin{cases} f_{t+1}^{\mathrm{src}} - f_t^{\mathrm{src}}, & \text{visual feature differences,} \\ \mathrm{IDM}(f_t^{\mathrm{src}}, f_{t+1}^{\mathrm{src}}), & \text{latent actions.} \end{cases} \tag{1}$$

We denote these two forms by $\Delta f_t^{\mathrm{src}}$ and z_t^{src}, respectively. Here, IDM includes the model-specific latent bottleneck, yielding either continuous latent actions or vector-quantized action tokens.

Transfer generation. Under the common decoding protocol, each representation uses a separately trained decoder G. Representation interfaces and decoder training are detailed in Section 4. Starting from the target initial frame, generation proceeds autoregressively:

$$\hat{I}_0^{\mathrm{tgt}} = I_0^{\mathrm{tgt}}, \qquad \hat{I}_{t+1}^{\mathrm{tgt}} = G\Big(\hat{I}_t^{\mathrm{tgt}}, r_t^{\mathrm{src}}\Big), \quad t = 0, \ldots, T-1. \tag{2}$$

The resulting sequence $\hat{V}^{\mathrm{tgt}} = (\hat{I}_t^{\mathrm{tgt}})_{t=0}^{T}$ should reproduce the source action effect while preserving the target context and avoiding source-appearance copying. LATBench evaluates these three requirements through AEF, TCP, and SCL, respectively.

4 LATBench

LATBench standardizes data construction, representation interfaces, conditional generation, and evaluation to compare models with different visual encoders, IDM inputs, and latent bottlenecks. This setting complements analyses of latent-action design choices in robot learning (Bu et al., 2026). Figure 1 outlines LATBench. During LAM pretraining, a frozen visual encoder supplies features to an IDM with a latent bottleneck and a forward dynamics model (FDM), which predicts features across augmented views. Pretraining is detailed in Appendix C. For transfer evaluation, we freeze the encoder and IDM and train a pixel decoder to generate the next frame from the IDM output and a reference frame. Unless otherwise specified, the pixel decoder refers to this configuration.

4.1 Benchmark Construction

We construct two evaluation suites from SSv2 (Goyal et al., 2017) and HRDexDB (Lim et al., 2026), denoted **SSv2 validation** and **HRDexDB validation**. SSv2 validation covers six directional

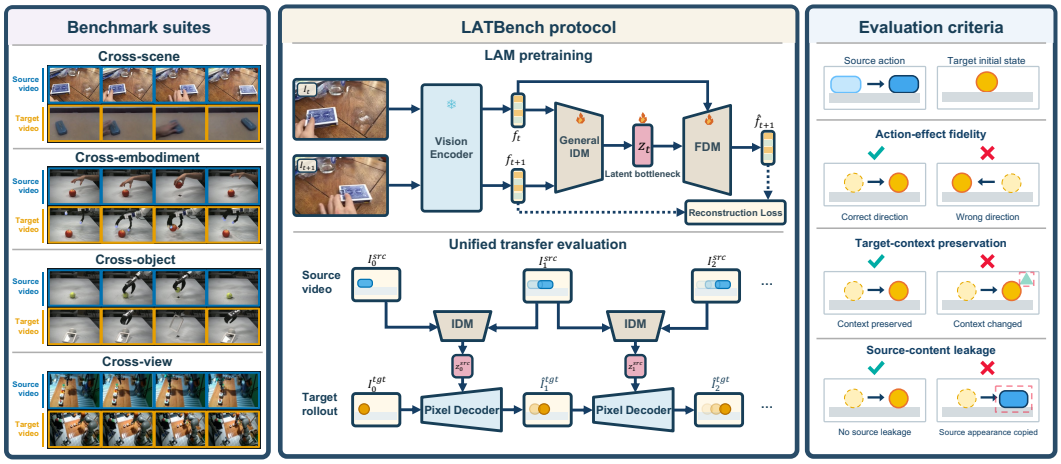


Figure 1: LATBench overview. Left: cross-scene, cross-embodiment, and cross-object transfer, with supplementary cross-view evaluation. Middle: LAM pretraining and source-to-target generation under a common decoding protocol. Right: AEF, TCP, and SCL.

object-motion actions in the image plane. HRDexDB validation covers manipulation by human and dexterous robotic hands. Their source–target pairings form cross-scene, cross-embodiment, and cross-object transfer settings. The LAM variants in Section 5.2 are pretrained on PE Video and Open X-Embodiment data, as detailed in Appendix C.1. Each suite samples source–target pairs from its base dataset without requiring matching action labels. Training and evaluation splits are disjoint at the source-video level, and directional effects are defined in a common image coordinate system. We also examine cross-view transfer qualitatively on RH20T (Section 5.4.3), as illustrated in Figure 1.

4.2 Unified Transfer Protocol

The primary evaluation decodes the source representations defined in Section 3, bypassing the FDM. We map each representation to a common conditioning interface and train a separate pixel decoder while keeping the representation extractor frozen. Decoder architecture, training data, optimization settings, and training budget are held fixed across all representations. The decoders therefore share a protocol but have separate weights. Appendix D details their architectures and training settings. Transfer rollouts start from the target initial frame, conditioned on source representations.

4.3 Evaluation

LATBench evaluates source-to-target rollouts using three vision-language models (VLMs): Gemini 3 Flash Preview (Google, 2026b), Gemini 3.5 Flash Lite (Google, 2026a), and DeepSeek V4 Flash Vision Exp (DeepSeek, 2026). Evaluators independently score AEF, TCP, and SCL given identical visual inputs and prompts. Each prefix contains $F = T+1$ frames, including the initial target frame. Appendix A details evaluator settings, frame sampling, prompts, and score aggregation.

Action-Effect Fidelity (AEF). Given the action description, the evaluator compares motion direction, temporal progression, and action completion between the source and generated prefixes. It assigns a single score $A_i \in [0, 1]$ to the full prefix of sample i. We report $\text{AEF} = N^{-1} \sum_{i=1}^{N} A_i$ over N samples. The scoring procedure and prompt are provided in Appendix A.3.

Target-Context Preservation (TCP). The evaluator assigns each predicted frame an identity-preservation score $p_{i,t} \in [0, 1]$ relative to the target reference at the same time step, allowing natural changes in pose and position. We compute the sample-level score $P_i = T^{-1} \sum_{t=1}^{T} p_{i,t}$ and report $\text{TCP} = N^{-1} \sum_{i=1}^{N} P_i$. On SSv2, TCP measures object and scene identity. On HRDexDB, it measures coarse human-versus-robot embodiment and object identity, excluding background similarity. The dataset-specific scoring procedures are detailed in Appendix A.4.

Table 1: Four-frame cross-context transfer of representative LAMs. Per-dataset values are mean ± SD across three evaluators. Avg. Transfer is the average of the two dataset means. Bold: best per-dataset metric or cross-dataset average.

Model	Dataset	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑	Avg. Transfer ↑
LAPA (Ye et al., 2025)	SSv2 validation	0.509 ± 0.044	0.487 ± 0.039	$\mathbf{0.022 \pm 0.010}$	0.245 ± 0.029	0.205
	HRDexDB validation	0.226 ± 0.143	0.609 ± 0.030	$\mathbf{0.003 \pm 0.002}$	0.165 ± 0.095	
UniVLA (Bu et al., 2025)	SSv2 validation	0.539 ± 0.052	0.636 ± 0.032	0.035 ± 0.029	0.336 ± 0.034	**0.310**
	HRDexDB validation	0.348 ± 0.120	$\mathbf{0.758 \pm 0.037}$	0.017 ± 0.014	$\mathbf{0.283 \pm 0.092}$	
villa-X (Chen et al., 2026b)	SSv2 validation	0.537 ± 0.047	0.615 ± 0.026	0.042 ± 0.036	0.321 ± 0.028	0.266
	HRDexDB validation	0.288 ± 0.143	0.605 ± 0.055	0.007 ± 0.005	0.211 ± 0.094	
LaWAM (Chen et al., 2026a)	SSv2 validation	$\mathbf{0.546 \pm 0.045}$	$\mathbf{0.674 \pm 0.033}$	0.046 ± 0.033	$\mathbf{0.355 \pm 0.037}$	0.306
	HRDexDB validation	$\mathbf{0.405 \pm 0.133}$	0.658 ± 0.020	0.125 ± 0.056	0.256 ± 0.054	

Source-Content Leakage (SCL). SCL detects source-specific appearance absent from the target context, excluding similarity attributable solely to motion or pose. The frame-level label $l_{i,t}$ is 1 for confirmed leakage, 0.5 for an unresolved hand or arm identity judgment without confirmed leakage, and 0 otherwise. The sample-level penalty is $L_i = \max_{1 \le t \le T} l_{i,t}$. For each evaluator, the dataset-level SCL is $\mathrm{SCL} = N^{-1} \sum_{i=1}^{N} L_i$, equivalently $\mathrm{SCL} = R_{\mathrm{confirmed}} + 0.5 R_{\mathrm{uncertain}}$, where $R_{\mathrm{confirmed}}$ and $R_{\mathrm{uncertain}}$ denote the confirmed and uncertain leakage rates, respectively. Screening, verification, and aggregation procedures are detailed in Appendix A.5.

Transfer Score. We compute $S_i = A_i P_i (1 - L_i)$ and report $S_{\mathrm{dataset}} = N^{-1} \sum_{i=1}^{N} S_i$. Target preservation continuously weights action fidelity, while source-content leakage imposes an additional penalty. A sample with confirmed leakage receives zero; an unresolved identity judgment without confirmed leakage halves the preservation-weighted action score. All four metrics are computed independently for each evaluator and reported as the mean ± standard deviation across three evaluators. Transfer Scores are averaged over samples before cross-evaluator aggregation, rather than computed from dataset-level component averages.

5 Experiments

We investigate cross-context action reuse through four questions: overall transfer performance of existing LAMs (RQ1), the effects of encoders, IDM inputs, and bottlenecks (RQ2), the distinction between action information and source content (RQ3), and failure modes under longer rollouts, camera motion, and viewpoint changes (RQ4).

Unless otherwise specified, all experiments follow the protocol in Section 4. LAM pretraining and pixel decoder settings are provided in Appendices C and D.

Baseline Models. We evaluate the LAMs from **LAPA**, **UniVLA**, **villa-X**, and **LaWAM**. For LAPA, we use the latent action quantization model, which learns discrete actions through vector quantization and future-frame reconstruction (Ye et al., 2025). For UniVLA, we use the second-stage LAM, which learns task-centric latent actions in DINOv2 feature space after a language-conditioned first stage that models task-irrelevant changes (Bu et al., 2025). For villa-X, we use the LAM module, trained through visual forward prediction with additional proprioceptive supervision when robot state and action annotations are available (Chen et al., 2026b). For LaWAM, we use the first-stage LAM, which infers continuous latent actions from frozen visual features and predicts future features (Chen et al., 2026a). We freeze all representation extractors and train separate pixel decoders under the common protocol, evaluating every model on the same source–target pairs.

5.1 RQ1: Overall Transferability of Existing Latent Actions

Table 1 shows that LaWAM leads on SSv2 in AEF, TCP, and Transfer Score. On HRDexDB, it achieves the highest AEF but also exhibits the most leakage, while UniVLA leads in TCP and Transfer Score. LAPA has the lowest leakage and Transfer Score on both datasets. All models have lower AEF on HRDexDB. Section 5.2 compares encoder, IDM, and bottleneck choices. Per-evaluator results and qualitative examples appear in Appendix Table 4 and Appendix I.

Table 2: Transfer Score for encoder–IDM combinations on SSv2 validation and HRDexDB validation (mean ± SD across three evaluators). Avg. Transfer is the average of the two dataset means for each IDM input type. Bold: highest mean per column. Full metrics: Appendix Table 13.

	SSv2		HRDexDB		Avg. Transfer ↑	
Encoder	LAPA-style	CoMo-style	LAPA-style	CoMo-style	LAPA-style	CoMo-style
DINOv3	$\mathbf{0.337 \pm 0.034}$	$\mathbf{0.361 \pm 0.045}$	$\mathbf{0.246 \pm 0.052}$	0.217 ± 0.035	**0.292**	0.289
V-JEPA 2.1	0.333 ± 0.022	0.354 ± 0.033	0.174 ± 0.068	$\mathbf{0.296 \pm 0.064}$	0.254	**0.325**
CLIP	0.321 ± 0.033	0.331 ± 0.026	0.161 ± 0.071	0.236 ± 0.040	0.241	0.284
MAE	0.294 ± 0.043	0.291 ± 0.046	0.229 ± 0.097	0.209 ± 0.108	0.262	0.250

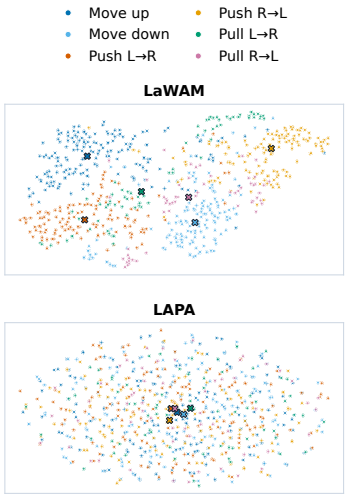


Figure 2: Independently fitted t-SNE projections on SSv2 validation. Crosses: coordinate-wise class medians.

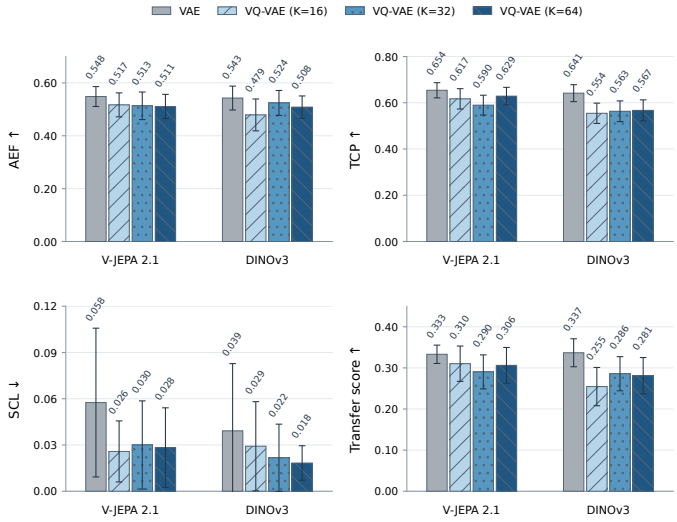


Figure 3: VAE versus VQ-VAE with LAPA-style IDMs on SSv2 validation. Bars: means; error bars: SD across three evaluators. Full results: Appendix Table 16.

We further examine the structure of video-level latent action representations using t-distributed stochastic neighbor embedding (t-SNE). Figure 2 reveals clearer action-class structure for LaWAM, while LAPA's classes overlap broadly. This difference offers a possible representation-level explanation for LaWAM's stronger transfer performance on SSv2. Appendix E details the feature extraction and dimensionality reduction procedures, along with complete results for all four models and comparisons of opposite actions.

Takeaway: Transfer quality depends on both the metric and the domain.

Existing LAMs exhibit trade-offs among action reproduction, target preservation, and source-content leakage, with no model achieving the highest Transfer Score in both evaluated domains.

5.2 RQ2: What Designs Determine Latent-Action Transferability?

We train LAM variants to examine the effects of the visual encoder, IDM design, and latent bottleneck, holding the training data, pixel decoder architecture, and optimization settings fixed. Complete encoder and IDM results are provided in Appendix Table 13.

Visual encoder. We combine DINOv3, V-JEPA 2.1, CLIP, and MAE with LAPA-style and CoMo-style IDMs (Table 2). With LAPA-style inputs, DINOv3 achieves the highest mean Transfer Score on both datasets. With CoMo-style inputs, DINOv3 leads on SSv2 and V-JEPA 2.1 on HRDexDB.

IDM design. The IDM variants share the same architecture but differ in their inputs: the LAPA-style variant uses features from adjacent frames, while the CoMo-style variant uses the current features and their temporal difference (Ye et al., 2025; Yang et al., 2026). The architecture and input formulations are detailed in Appendix C.3. Averaged over encoders, CoMo-style improves Transfer Score on both datasets. The effect varies by encoder, however: V-JEPA 2.1 and CLIP improve on both datasets, MAE declines on both, and DINOv3 improves only on SSv2. V-JEPA 2.1 with CoMo-style IDM inputs achieves the highest average Transfer Score across the two datasets (0.325), making it the preferred default among the evaluated encoder–IDM combinations.

These differences may reflect how encoder pretraining interacts with the data domain. V-JEPA 2.1 learns spatiotemporal representations through video prediction, so differences between its frame features may more readily capture action-related changes (Mur-Labadia et al., 2026). CLIP is trained with an image–text contrastive objective and no explicit temporal supervision; difference inputs may help its IDM extract state changes (Radford et al., 2021). Both encoders benefit from CoMo-style inputs, albeit with different trade-offs. On HRDexDB, V-JEPA 2.1 improves action fidelity and target preservation while reducing leakage, whereas the gains of CLIP in action fidelity come with increased leakage. The pixel reconstruction objective of MAE instead emphasizes local appearance and spatial detail (He et al., 2022). Differencing may accentuate texture and occlusion changes in these features, while paired frame features may make cross-frame correspondences easier to learn.

DINOv3 further illustrates this dependence on the data domain. Its self-supervised representations retain semantic and spatial structure without explicitly separating action from embodiment-specific features (Siméoni et al., 2025). Our SSv2 subset contains diverse scenes and mainly directional manipulations, where difference inputs may highlight relative state changes. HRDexDB has consistent backgrounds, with changes concentrated on the hands and manipulated objects. Here, differencing may emphasize both the action and the source hand's morphology. This interpretation is consistent with the higher AEF, lower TCP, and higher SCL observed for CoMo-style inputs on HRDexDB.

Latent information bottleneck. We compare continuous VAE and discrete VQ-VAE bottlenecks on SSv2 validation using a fixed LAPA-style IDM, with codebook size $K \in \{16, 32, 64\}$ (Kingma & Welling, 2014; Van Den Oord et al., 2017). We evaluate DINOv3 and V-JEPA 2.1 with the bottleneck settings described in Appendix C.4.

For both encoders, every VQ-VAE configuration yields lower mean SCL, AEF, TCP, and Transfer Score than its VAE counterpart (Figure 3). TCP declines even though the generator still receives the target image. This suggests that preserving the target context also depends on whether the action representation supports the required state changes.

Takeaway: Encoder features, IDM inputs, and latent bottlenecks must jointly support action reuse.

Encoder quality for transfer depends on how the IDM uses its features. V-JEPA 2.1 with CoMo-style inputs is the preferred default. Explicit differencing does not consistently improve transfer, while discretization reduces leakage at the cost of action fidelity and target preservation.

5.3 RQ3: What Is Transferred: Action or Source-Specific Visual Content?

To test whether transferring motion also transfers source appearance, we compare DINOv2 and DINOv3 feature differences (Oquab et al., 2023; Siméoni et al., 2025) with LAPA-DINOv2 and LAPA-DINOv3 from LARYBench (Nie et al., 2026). Each LAM uses the corresponding DINO encoder, with an IDM and latent bottleneck learned through LAPA-style training (Ye et al., 2025). All four representations use the same source–target pairs and the decoding protocol described in Section 4. Table 3 reports four-frame results on SSv2 validation, and Appendix Table 12 provides the evaluator-specific measurements.

DINO feature differences encode motion that the decoder can reproduce, but they also carry source appearance into the target rollout (Figure 4). LAPA-style training reduces SCL from 0.536 and 0.435 to 0.045 for both encoders, with only modest losses in AEF. TCP shifts in opposite directions for the two encoders, yet Transfer Score improves in both cases.

Table 3: Four-frame transfer of visual feature differences versus LAPA-style latent actions on SSv2 validation (mean ± SD across three evaluators).

Representation	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
DINOv2 (Oquab et al., 2023)	0.601 ± 0.055	0.661 ± 0.009	0.536 ± 0.137	0.190 ± 0.057
DINOv3 (Siméoni et al., 2025)	$\mathbf{0.605 \pm 0.007}$	0.669 ± 0.014	0.435 ± 0.178	0.228 ± 0.071
LAPA-DINOv2 (Nie et al., 2026)	0.563 ± 0.037	$\mathbf{0.677 \pm 0.038}$	$\mathbf{0.045 \pm 0.036}$	0.365 ± 0.035
LAPA-DINOv3 (Nie et al., 2026)	0.586 ± 0.033	0.657 ± 0.034	$\mathbf{0.045 \pm 0.039}$	$\mathbf{0.374 \pm 0.032}$



Figure 4: Source-content leakage in transfer from a black object to a white bulb. DINOv2/v3 copy source-object and hand appearance; LAPA-DINOv2/v3 preserve the target object and background.

Takeaway: Latent-action learning suppresses SCL while largely preserving AEF.

Compared with visual feature differences from the same encoder, learned latent actions substantially reduce source-appearance copying while retaining most of the ability to reproduce action effects, leading to better overall transfer.

5.4 RQ4: Failure Modes of Latent-Action Transfer

We further examine error accumulation in longer rollouts and transfer failures under camera motion and viewpoint changes.

5.4.1 Error Accumulation in Long-Horizon Rollouts

At each rollout step, the decoder receives the previous prediction together with the corresponding source latent action, allowing errors to accumulate. We evaluate four LAMs on SSv2 validation using prefixes of 2, 4, 6, and 8 frames from the same canonical sequence, including the initial frame. The frame sampling procedure is described in Appendix A.2. In Figure 5, AEF initially rises as more motion becomes visible, then plateaus or declines. TCP decreases and SCL increases with rollout length; beyond four frames, Transfer Score falls for every model. Complete aggregate and evaluator-specific results are provided in Appendix G. The main experiments therefore use four-frame prefixes to balance AEF, TCP, and SCL.

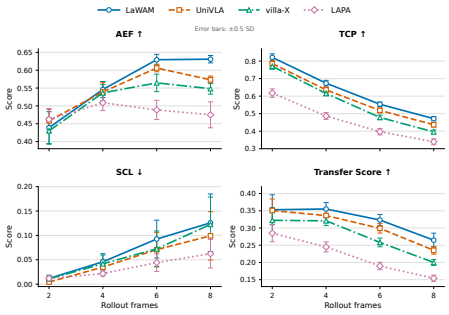


Figure 5: Transfer across rollout lengths on SSv2 validation. Means across three evaluators; error bars: ±0.5 SD.

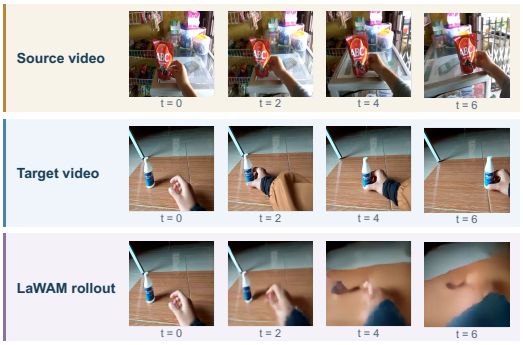


Figure 6: LaWAM under large camera motion. Rows: source, target reference, and generated videos ($t = 0, 2, 4, 6$).

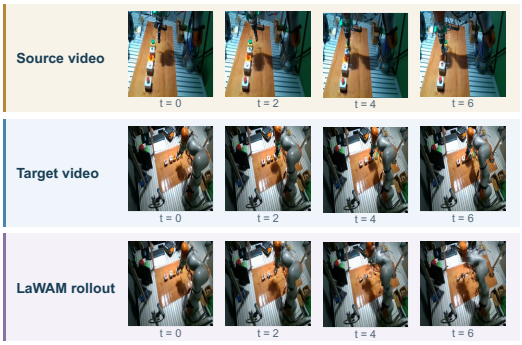


Figure 7: Cross-view transfer with LaWAM. Rows: source, target reference, and generated videos ($t = 0, 2, 4, 6$).

5.4.2 Large Camera Motion

We use an SSv2 source video with pronounced camera motion to test whether latent actions can transfer object motion while preserving the target scene. In Figure 6, the scene generated by LaWAM shifts and tilts, the target bottle loses its shape, and the hand becomes distorted. This example suggests that latent actions can encode camera motion along with object motion, introducing unwanted global changes into the target scene. Source-camera motion shifts the image positions of both foreground and background, making it harder to isolate object motion from the observed inter-frame changes. Appendix Figure 13 shows five additional cases with different objects and scenes.

5.4.3 Cross-View Transfer

We train the pixel decoder on a small subset of RH20T videos to evaluate cross-view action transfer (Fang et al., 2024). At test time, we extract latent actions from a robot manipulation video in one view and apply them to an initial image from another view. The same physical action can produce different image motions across viewpoints, so cross-view transfer requires mapping the source action to visual changes in the target view. In Figure 7, LaWAM initially preserves the target scene, but the robot arm later develops ghosting and deformation, and tabletop objects become blurred. The rollout does not clearly reproduce the manipulation from the target view. These results indicate limited adaptation to viewpoint-dependent motion under this training setup. Appendix Figure 13 shows five further cross-view cases.

Takeaway: Action reuse remains limited across time and visual conditions.

Longer rollouts expose declining target preservation and increasing leakage, while camera-motion and cross-view examples reveal difficulties in adapting source actions to changing visual conditions.

6 Conclusion

LATBench evaluates whether latent actions can reproduce source action effects in a different visual context while preserving the target object and scene. Its unified protocol jointly measures AEF, TCP, and SCL. Our comparisons show that learned latent actions improve transfer by suppressing source content while retaining most of the action effect. Discretization also reduces leakage, but can impair action reproduction and target preservation. Effective transfer therefore depends on what information is retained, not simply on how much source content is removed. Encoder features, IDM inputs, and latent bottlenecks must be considered together, while longer rollouts and qualitative camera-motion and cross-view tests reveal further limits to action reuse. These findings motivate the design and evaluation of latent actions for cross-context reuse, with joint attention to action reproduction, target preservation, and source-content suppression.

AI USAGE STATEMENT

In this work, we used generative AI tools for dataset construction and automated evaluation. Additionally, we used generative AI tools for writing code and polishing the manuscript to improve grammar, clarity, and readability. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of AI.

REPRODUCIBILITY STATEMENT

We provide detailed experimental specifications in the appendix to support reproducibility. Appendix B describes dataset construction and splits. Appendices C and D detail the training data, network architectures, loss functions, and optimization settings for LAM pretraining and pixel decoder training, respectively. Appendix A specifies the evaluation protocols, including frame sampling, evaluator prompts, and score aggregation. We will publicly release the code and experimental configurations upon acceptance.

REFERENCES

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026.

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Shang-Wen Li, Piotr Dollár, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. In *Advances in Neural Information Processing Systems*, 2025.

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

Xizhou Bu, Qingda Hu, Lei Zhou, Lingfeng Zhang, Yingbo Tang, Zihao Liu, Xinyi Tao, Zhiqiang Ma, Qingqiu Huang, Chufeng Tang, et al. What matters for latent actions in robot learning. *arXiv preprint arXiv:2608.19613*, 2026.

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*, 2026a.

Xiaoyu Chen, Junliang Guo, Tianyu He, Chuheng Zhang, Pushi Zhang, Derek Cathera Yang, Li Zhao, and Jiang Bian. Igor: Image-goal representations are the atomic control units for foundation models in embodied ai. *arXiv preprint arXiv:2411.00785*, 2024.

Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, et al. Villa-x: enhancing latent action modeling in vision-language-action models. In *International Conference on Learning Representations*, volume 2026, pp. 70673–70703, 2026b.

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *IEEE/CVF International Conference on Computer Vision*, pp. 19752–19763. IEEE, 2025.

DeepSeek. DeepSeek-V4-Flash-Vision-Exp release: Multimodal API now live. DeepSeek API Docs, August 2026. URL `https://api-docs.deepseek.com/news/news260821/`.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. RH20T: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 653–660. IEEE, 2024.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. In *Forty-second International Conference on Machine Learning*, volume 267, 2025.

Quentin Garrido, Tushar Nagarajan, Basile Terver, Nicolas Ballas, Yann LeCun, and Michael Rabbat. Learning latent action world models in the wild. In *Forty-third International Conference on Machine Learning*, 2026.

Google. Gemini 3.5 Flash-Lite. Google AI for Developers, 2026a. URL `https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash-lite`.

Google. Gemini 3 Flash. Google Cloud Documentation, 2026b. URL `https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/gemini/3-flash`.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 5843–5851. IEEE, 2017.

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

Yuxin Jiang, Yuchao Gu, Ivor W Tsang, and Mike Zheng Shou. Olaf-world: Orienting latent actions for video world modeling. In *Forty-third International Conference on Machine Learning*, 2026.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Anthony Liang, Pavel Czempin, Matthew M Hong, Yutai Zhou, Jingzhen Wang, Erdem Biyik, and Stephen Tu. Clam: Continuous latent action models for robot learning from unlabeled demonstrations. *arXiv preprint arXiv:2505.04999*, 2025.

Jongbin Lim, Taeyun Ha, Mingi Choi, Jisoo Kim, Byungjun Kim, Subin Jeon, and Hanbyul Joo. Hrdexdb: A paired human-robot dataset for cross-embodiment dexterous grasping. *arXiv preprint arXiv:2604.14944*, 2026.

Jing Lyu, Shuanghao Bai, Runze Xiao, Zhenyu Liao, Wenxing Tan, Zihan Tang, Ruochuan Shi, Cheng Peng, Yuheng Ji, Yihao Wang, Badong Chen, Pengwei Wang, Zhongyuan Wang, and Xiaoguang Zhao. DIDO: Distilling interaction-centric dynamics into one-step denoising for world action models. *arXiv preprint arXiv:2609.15570*, 2026. arXiv:2609.15570v2.

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.

Dujun Nie, Fengjiao Chen, Qi Lv, Jun Kuang, Xiaoyu Li, Xuezhi Cao, and Xunliang Cai. Lary: A latent action representation yielding benchmark for generalizable vision-to-action alignment. *arXiv preprint arXiv:2604.11689*, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, et al. Open x-embodiment: Robotic learning datasets and RT-X models : Open x-embodiment collaboration. In *IEEE International Conference on Robotics and Automation*, pp. 6892–6903. IEEE, 2024.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Sandeep Kumar Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. Vipra: Video prediction for robot actions. In *International Conference on Learning Representations*, volume 2026, pp. 68823–68857, 2026.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations*, volume 2024, pp. 9379–9395, 2024.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in neural information processing systems*, volume 30, 2017.

Junke Wang, Qihang Zhang, Shuai Yang, Yiming Luo, Yujun Shen, Zuxuan Wu, Yu-Gang Jiang, and Yinghao Xu. Repwam: World action modeling with representation visual-action tokenizers. *arXiv preprint arXiv:2606.13674*, 2026.

Jiange Yang, Yansong Shi, Haoyi Zhu, Mingyu Liu, Kaijing Ma, Yating Wang, Gangshan Wu, Tong He, and Limin Wang. Como: Learning continuous latent motion from internet videos for scalable robot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 42352–42363, 2026.

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

Hu Yue, Siyuan Huang, Yue Liao, Shengcong Chen, Pengfei Zhou, Liliang Chen, Maoqing Yao, and Guanghui Ren. Ewmbench: Evaluating scene, motion, and semantic quality in embodied world models. *arXiv preprint arXiv:2505.09694*, 2025.

Tianqiu Zhang, Muyang Lyu, Yufan Zhang, Fang Fang, and Si Wu. DiLA: Disentangled latent action world models. In *Forty-third International Conference on Machine Learning*, 2026.

A AUTOMATIC EVALUATION PROTOCOL FOR GENERATED VIDEOS

This appendix supplements Section 4.3 with details on evaluator inputs, frame sampling, scoring procedures, and the prompts used for automatic evaluation.

A.1 EVALUATOR AND INPUT SETTINGS

We use three vision-language models (VLMs) to evaluate action-effect fidelity (AEF), target-context preservation (TCP), and source-content leakage (SCL): Gemini 3 Flash Preview (Google, 2026b), Gemini 3.5 Flash Lite (Google, 2026a), and DeepSeek V4 Flash Vision Exp (DeepSeek, 2026). Each evaluator runs the full protocol independently on the same samples, using identical prompts and visual inputs. We score each dimension in separate calls with different inputs. AEF uses the source and generated video prefixes. TCP uses a target reference frame and the corresponding generated frame. SCL uses source and target references together with the generated frame, with local crops added during verification.

Candidates are anonymous, and generation model names are withheld. AEF calls do not receive the target reference video, and TCP calls do not receive the source video.

A.2 FRAME SAMPLING AND EVALUATION HORIZON

We uniformly sample eight canonical frames from the full duration of each generated video, indexed by $t = 0, \ldots, 7$. We select the corresponding frames from the source and target reference videos. For each prefix length $F \in \{2, 4, 6, 8\}$, we take the first F canonical frames without resampling the full video. Four-frame evaluation, for example, uses frames $0, 1, 2, 3$.

Frame $t = 0$ is the target initial frame supplied to the generator, $\hat{I}_0^{\mathrm{tgt}} = I_0^{\mathrm{tgt}}$. A prefix therefore contains $T = F - 1$ predicted frames, consistent with the notation in Sections 3 and 4.3. AEF evaluates the entire prefix, including the initial frame. TCP and SCL evaluate only predicted frames $t = 1, \ldots, T$. The main experiments use four-frame prefixes comprising one initial frame and three predicted frames. In the prompts below, `t0`, ..., `t{last_frame}` denote the canonical frames visible in the evaluated prefix, where `last_frame` equals $F - 1$. Frames beyond this index are neither shown to the evaluator nor assumed during scoring.

A.3 ACTION-EFFECT FIDELITY (AEF)

AEF measures how well the generated video executes the source action. The evaluator receives separate chronological frame grids for the source and generated prefixes, together with the action description for the sample. It returns a video-level score $A_i \in [0, 1]$ based on visible motion direction, temporal progression, and action completion. The score is not an average of frame-level action scores; object identity, color, texture, and background similarity are excluded from the criteria.

The prompt is:

```
Evaluate ONLY execution of the intended action in one anonymous
candidate video prefix. Image 1 is the SOURCE motion reference. Image
2 is the anonymous CANDIDATE. Each image contains the same
chronological prefix of {frame_count} frames.
{layout_description}
Judge only visible evidence through t{last_frame}; do not assume
unshown future motion. The intended action is: {action}. Judge visible
motion direction, temporal progression, and completion of this action.
Object identity, color, texture, background, and visual similarity are
not criteria. Judge against the stated action rubric, not against
another candidate or a generation method. Use [0,1]: 0 = absent or
contradictory action; 0.25 = weak/inconclusive motion; 0.50 =
recognizable but substantially incomplete action; 0.75 = mostly
correct action with limited completion; 1 = clear correct direction
and completed action. Intermediate values are allowed. These anchors
are a rubric, not probabilities. Return ONLY JSON exactly like
```

```
{"action_score":0.0}, replacing the example with one finite numeric
value in [0,1]. No boolean, explanation, or other score.
```

The placeholders {frame_count}, {layout_description}, {last_frame}, and {action} are replaced with the input frame count, grid layout, last visible frame index, and sample-specific action description, respectively.

A.4 TARGET-CONTEXT PRESERVATION (TCP)

We evaluate TCP separately at each predicted time step. The target reference and generated frames are placed side by side, with the reference on the left and the candidate on the right, and scored in a separate call. The reference establishes target identity and appearance; it does not prescribe a motion trajectory or require pixel alignment.

We average the frame scores $p_{i,t}$ to obtain the sample-level preservation score P_i. The tables report TCP as the mean of P_i across evaluation samples, on the $[0, 1]$ scale.

A.4.1 SSv2

On SSv2, TCP assesses whether the target object's identity, characteristic colors, textures, and scene background remain recognizable. Natural changes in position, pose, viewpoint, or occlusion do not by themselves constitute identity loss.

The prompt is:

```
Evaluate ONLY preservation of TARGET appearance in one anonymous
same-time frame pair. This call contains exactly one image. The TARGET
REFERENCE at time t is on the left and the anonymous CANDIDATE at the
same time t is on the right. Judge whether the target object's
identity, characteristic colors/textures, and scene/background
identity remain recognizable in the candidate. Natural changes of
position, pose, viewpoint and occlusion do not by themselves imply
identity loss. Do not use another time step, score action execution,
or compare with another candidate. Use [0,1]: 0 = target identity
absent/replaced; 0.25 = weak fragments retained; 0.50 = partly
retained with substantial identity loss; 0.75 = mostly retained with
minor changes; 1 = consistently preserved. Intermediate values are
allowed. Return ONLY JSON exactly like {"target_identity_score":0.0},
replacing the example with one finite numeric value in [0,1]. No
boolean, explanation, or other score.
```

A.4.2 HRDEXDB

Source and target videos in HRDexDB often show the same manipulated object, background, and camera viewpoint. Background similarity therefore provides no evidence of target preservation and is excluded from scoring. The evaluation focuses on the target embodiment (human versus robot) and the manipulated object.

The rubric allows pose changes, local blur, and finger deformation as long as the target embodiment remains recognizable. Intermediate scores cover cases in which the coarse embodiment appears unchanged but occlusion or deformation makes finer details difficult to identify.

The prompt is:

```
Evaluate ONLY preservation of TARGET visual identity in one anonymous
cross-embodiment frame pair. This call contains exactly one image. The
TARGET REFERENCE is on the left and the anonymous CANDIDATE at the
same time is on the right.

The required target embodiment is: {target_embodiment}.
```

```
This benchmark transfers motion between a human hand and different
robot hands. Source and target often use the same object, camera and
background. Ignore the background: it cannot establish target
preservation. Preserving only the object or table is NOT sufficient.
Check whether the right side still shows the same target
human-versus-robot embodiment, arm/hand or gripper morphology,
materials and characteristic colors as the left. Also check that the
manipulated target object has not been replaced.

Judge recognizable identity, not exact pose, action timing, pixel
alignment, or image quality. A blurry, articulated, partly occluded,
or mildly deformed hand still counts as the target when it is visibly
the same human/robot embodiment. Do not penalize an occlusion that
also occurs on the left.

First decide the coarse embodiment from visible evidence. For a robot
target, a connected robot arm or end effector with target-consistent
materials counts as retained even if generated fingers are malformed
or fewer fingers are visible. For a human target, a visible human arm
or hand counts as retained even if fingers are blurred or malformed.
Do not call the target absent merely because exact finger geometry
cannot be verified.

Return 1 when the same coarse target embodiment is clearly visible and
the target object is not replaced. Return 0.6 when the coarse
embodiment is plausibly the same and the object is not clearly
replaced, but heavy blur, deformation, crop or occlusion prevents
confident verification or the exact robot family is uncertain. Return
0.5 only when the coarse embodiment is plausibly the same but object
identity is also uncertain. Return 0 only for a clear
human-versus-robot replacement, complete absence of the target
arm/hand, or a clearly replaced target object. A visible target-type
arm/end effector together with the same object forbids a zero. Do not
score action execution and do not output any component score. Return
ONLY JSON exactly like {"target_identity_score":0.0}, replacing the
example with one finite numeric value in [0,1]. No boolean,
explanation, or other score.
```

The placeholder {target_embodiment} is replaced according to sample metadata:

```
human:
the human arm and five-finger hand shown in the target reference

allegro_v5:
the Allegro V5 robot arm and black multi-finger gripper shown in the
target reference

inspire_f1:
the Inspire F1 robot arm and dexterous hand shown in the target
reference

inspire_dftp:
the Inspire DFTP robot arm and dexterous hand shown in the target
reference
```

A.5 SOURCE-CONTENT LEAKAGE (SCL)

SCL detects source-specific appearance in generated frames that the target references cannot explain. This includes objects, textures, markings, sleeves, background elements, and local structure. Action-induced similarities in pose, grasp, motion direction, or trajectory do not count as leakage.

Evaluation follows a cascade of checks for each predicted frame. Localization-based screening first identifies possible source-specific appearance. Positive frames then undergo independent verification. Proposed hand and arm regions receive an additional identity review to distinguish copied appearance from similarity caused by the transferred pose.

A frame is labeled as confirmed leakage if any region passes all required verification stages. If no region is confirmed but the identity of a proposed hand or arm region cannot be resolved, the frame is labeled uncertain. All remaining frames are labeled as no detected leakage.

A.5.1 Localization-Based Screening

Screening uses the current source frame, the target initial frame, the current target reference frame, the current generated frame, a temporal grid of exactly the F visible source-prefix frames, and a grayscale comparison of the current source and generated frames. The source grid includes each frame indexed by $t = 0, \ldots, F-1$ exactly once, with no additional temporal resampling. The layout is 1×2, 1×4, 2×3, or 2×4 for $F = 2, 4, 6, 8$, respectively. Frames are resized without changing their aspect ratios. The grid helps identify source appearance copied from a different time step, but only the current generated frame is scored.

The screening prompt is:

```
Audit SOURCE APPEARANCE leakage in one anonymous candidate frame.
Image 1: source frame at the corresponding time (appearance must not
be copied). Image 2: target initial frame (appearance to preserve).
Image 3: target reference frame at the corresponding time. Image 4:
anonymous candidate CENTER frame, the ONLY frame to score. Image 5:
chronological visible source-prefix frames t0..t{last_frame}, to
locate source evidence if shifted in time. Image 6: grayscale versions
of image 1 and image 4, as additional shape evidence. All standalone
frames use the same display size with aspect ratio preserved.

Inspect the entire candidate, including small patches,
hand/fingers/wrist/arm, sleeve, held objects, dark rectangles,
textures, and background. A copied source hand or arm IS appearance
leakage when its visible appearance or distinctive local structure
matches source and differs from target. Examine contour and
proportions together with wrist/sleeve boundary, markings, texture,
occlusions, and hand-object arrangement. Do not exclude people or
hands from inspection. Recoloring, resizing, partial copying, blur,
and changes in brightness do not erase identity. Require identifiable
source-specific visual evidence absent from target; a generic hand,
matching pose/motion/direction, skin color alone, or target
degradation alone is insufficient. Identical action or pose is allowed
when the target hand/object appearance is retained. Inspect actual
local structure rather than declaring leakage just because images look
alike. An artifact already present in target is not source-exclusive
leakage. If evidence is insufficient, return no detected leakage and
low confidence; do not invent an object or infer identity from model
quality.

Score only image 4. If any region visibly copies source-exclusive
appearance, set leakage_present=true and localize up to 4 strongest
matching regions in both original (unpadded) source and candidate
coordinates. EVERY bbox must be four FRACTIONS between 0.0 and 1.0 in
[x_left, y_top, x_right, y_bottom] order; x_left < x_right and y_top <
y_bottom. Example: [0.12,0.35,0.62,0.95]. Never output pixel
coordinates or numbers in the 0..1000 scale. The source index must
refer to t0..t{last_frame} in image 5. Descriptions must state the
concrete shared visual feature and why target does not explain it. Do
not describe motion as the matching feature. Otherwise regions must be
empty. Confidence is confidence in the boolean decision, not leakage
severity. Use short English evidence. Return only JSON, no reasoning
outside JSON:
{"leakage_present":false,"confidence":0.0,"regions":[]}
```

```
For a positive result, use the following region structure:
{"leakage_present":true,"confidence":0.9,"regions":[
{"category":"hand_appearance","source_frame_index":0,
"source_bbox":[0.12,0.35,0.62,0.95],
"candidate_bbox":[0.10,0.30,0.65,0.90],
"evidence":"shared appearance; difference from target"}]}

PREFIX_FRAME_INDEX_CONSTRAINT: Image 5 contains exactly {frame_count}
visible source-prefix frames t0..t{last_frame}. No source frame after
t{last_frame} is shown or may be assumed. Every source_frame_index
must be an integer from 0 to {last_frame}. Never cite a hidden source
frame. If leakage cannot be localized to one visible source frame,
return {"leakage_present":false,"confidence":0.5,"regions":[]}.
```

A.5.2 INDEPENDENT VERIFICATION

Independent verification examines the full generated frame again and localizes source-specific appearance without assuming that the screening decision is correct. The evaluator looks for concrete visual matches beyond action or pose and then checks whether the target references can explain them.

The prompt is:

```
Assess source appearance copying in one anonymous generated frame.
Image 1 is SOURCE at the current time. Image 2 is TARGET initial
appearance. Image 3 is TARGET at the corresponding time. Image 4 is
the CANDIDATE. Image 5 is the visible chronological source prefix
t0..t{last_frame}, containing exactly {frame_count} frames.

The source provides the desired ACTION. It is therefore normal for the
candidate to share source hand pose, finger placement, grasp
configuration, orientation, trajectory, and position. These
action-related similarities must NOT establish source appearance
copying, even when target's actual hand pose is different. First
consider whether the candidate can be explained by the TARGET
hand/object performing that action, with ordinary decoder blur or
shape distortion.

Count leakage only when a SOURCE-SPECIFIC appearance actually appears:
a copied object or object fragment; source-specific
outline/proportions distinct from pose; skin/texture/marking pattern;
cuff, jewelry or sleeve design; background structure; or an additional
source-like hand/arm overlaid on an otherwise retained target
hand/scene. An extra hand must have visual correspondence to source,
not just be a generic decoder artifact. Hands and arms are eligible,
but "same pose", "same finger placement", "same position", "same
motion", generic skin tone, and blur alone are insufficient.
Recoloring does not exempt a copied source object or hand when
distinctive structure/appearance remains. Compare against both target
images; target disappearance alone is not evidence of source copying.

Inspect the full candidate anew. Do not assume there is a match. If
there is clear source-exclusive appearance evidence, localize it
independently and state the shared visual feature beyond pose and why
target appearance cannot explain it. Otherwise leakage_present=false
with empty regions. Confidence describes the boolean judgment, not
severity. Do not name the model or infer its quality. Return ONLY JSON
with exactly these fields:
{"leakage_present":false,"confidence":0.95,"regions":[]}
For positive results, regions contains 1..4 objects with EXACTLY these
keys:
{"category":"hand_appearance","source_frame_index":3,
"source_bbox":[0.1,0.2,0.5,0.8],
```

```
"candidate_bbox":[0.2,0.3,0.6,0.9],
"evidence":"concrete source-exclusive appearance beyond action/pose"}
Allowed categories: hand_appearance, arm_sleeve, object, background,
other_appearance. EVERY box uses fractions from 0 to 1 in
x_left,y_top,x_right,y_bottom order in the unpadded original image;
x_left<x_right, y_top<y_bottom. Never output 0..1000 or pixel
coordinates. The source frame index must be an integer from 0 to
{last_frame}.

PREFIX_FRAME_INDEX_CONSTRAINT: Image 5 contains exactly {frame_count}
visible source-prefix frames t0..t{last_frame}. No source frame after
t{last_frame} is shown or may be assumed. Every source_frame_index
must be an integer from 0 to {last_frame}. Never cite a hidden source
frame. If leakage cannot be localized to one visible source frame,
return {"leakage_present":false,"confidence":0.5,"regions":[]}.
```

A.5.3 HAND AND ARM IDENTITY REVIEW

For proposed hand and arm regions, the evaluator receives full images, candidate-region crops, local target windows, and the visible source-prefix sequence. This review assesses hand or arm appearance only; similarity of a nearby object does not establish a hand identity match.

The output is `confirmed`, `not_supported`, or `uncertain`. The evaluator returns `uncertain` when blur, occlusion, or lack of target visibility prevents it from assessing source-specific details. Matching gestures, grasps, generic skin tone, or generation artifacts alone are insufficient to confirm leakage.

The prompt is:

```
Review a proposed HAND/ARM appearance match in an anonymous candidate.
The proposal is NOT ground truth. Previous scores, model names, and
previous explanations are deliberately withheld. Images 1..4: full
SOURCE proposed time, TARGET initial, TARGET current, CANDIDATE
current. Image 5: proposed SOURCE local crop. Image 6: proposed
CANDIDATE local crop. Images 7..10: four overlapping local windows
covering TARGET current, in TL, TR, BL, BR order. These are spatial
windows, NOT aligned identity matches. A hand can move to a different
window. Use full TARGET initial/current and the visible source-prefix
sequence (last image) to avoid false spatial assumptions. All crops
preserve aspect ratio; upscaling does not create visible detail.

The desired action is provided by SOURCE. Same pose, grasp, finger
placement, position, direction, silhouette due to pose, generic skin
tone, blur, extra fingers or ghost hands alone MUST NOT establish
appearance copying. An extra hand needs identifiable source-specific
appearance, not merely an artifact. Require a visible distinctive
source feature actually reproduced in candidate, such as a specific
cuff/jewelry/marking/texture or distinctive local detail independent
of pose. Recoloring alone does not rule copying out. Inspect only the
proposed HAND/ARM appearance, NOT a nearby held object or background.
Do not use an object's similarity as evidence for copying the hand.

confirmed: a visible source-specific non-pose feature is reproduced,
and target appearance cannot explain it. Specify the feature and
explicit target contrast. not_supported: the alleged correspondence is
just action/generic appearance or candidate appearance is explained by
target. This is not proof of global cleanliness. uncertain:
source-specific appearance cannot be resolved because of blur,
occlusion, missing target visibility, or ambiguous identity. Never
force uncertainty to true or false. Confidence is not a calibrated
probability or severity. Additionally report identity_assessable as a
native JSON boolean. It MUST be false when blur, occlusion, missing
target visibility, or ambiguous identity prevents judging the proposed
```

```
source-specific detail. Merely not seeing a detail in a blurred image
is not evidence of its absence. If identity_assessable=false, status
MUST be uncertain. When it is true, use confirmed or not_supported
according to the rules above. Generic pose-only similarity can be
not_supported when the images are sufficiently clear; no distinctive
matching feature is required to reject a clearly unsupported proposal.
Return ONLY JSON with EXACT keys:
{"status":"uncertain","identity_assessable":false,"confidence":0.5,
"source_specific_feature":"none visible",
"target_comparison":"target hand occluded",
"reason":"insufficient identity detail"}
status must be confirmed, not_supported, or uncertain. Each text field
must be nonempty and concise. confidence must be a finite numeric
value in [0,1].

VISIBLE_SOURCE_PREFIX: The last image contains exactly {frame_count}
visible source-prefix frames t0..t{last_frame}. No source frame after
t{last_frame} is shown or may be assumed. The proposed source frame is
guaranteed to lie within the visible index range 0..{last_frame}.
```

Across the screening, independent-verification, and hand/arm-review prompts, {`frame_count`} and {`last_frame`} are replaced by F and $F-1$, respectively. Each evaluator therefore receives only the source frames visible under the selected evaluation horizon.

A.5.4 Label Aggregation

The frame-level leakage label is

$$l_{i,t} = \begin{cases} 1, & \text{at least one region is confirmed as source-content leakage,} \\ 0.5, & \text{no region is confirmed, but a hand or arm region is uncertain,} \\ 0, & \text{otherwise.} \end{cases}$$

The sample-level leakage penalty is $L_i = \max_{1 \le t \le T} l_{i,t}$, where $T = F - 1$. Any confirmed future frame sets $L_i = 1$. If no frame is confirmed but at least one is uncertain, $L_i = 0.5$; otherwise, $L_i = 0$. These labels and penalties are computed independently for each evaluator. For evaluator e, let $R^{(e)}_{\text{confirmed}}$ and $R^{(e)}_{\text{uncertain}}$ denote the proportions of samples with $L^{(e)}_i = 1$ and $L^{(e)}_i = 0.5$, respectively. The reported evaluator-specific leakage rate is

$$\text{SCL}^{(e)} = R^{(e)}_{\text{confirmed}} + 0.5 R^{(e)}_{\text{uncertain}} = \frac{1}{N} \sum_{i=1}^{N} L^{(e)}_i.$$

A.6 Combined Score and Reporting

Following Section 4.3, the sample-level Transfer Score combines AEF, continuous target preservation, and the leakage penalty: $S^{(e)}_i = A^{(e)}_i \, P^{(e)}_i \, (1 - L^{(e)}_i)$. For each evaluator e, the dataset-level Transfer Score is the arithmetic mean of $S^{(e)}_i$ over all N samples. AEF, TCP, SCL, and Transfer Score are first computed as evaluator-specific dataset metrics. For any such metric M, we report the population mean and standard deviation across the three evaluators:

$$\bar{M} = \frac{1}{3} \sum_{e=1}^{3} M^{(e)}, \qquad \sigma_M = \sqrt{\frac{1}{3} \sum_{e=1}^{3} \left(M^{(e)} - \bar{M} \right)^2},$$

and display each result as $\bar{M} \pm \sigma_M$. Table 4 provides the evaluator-specific measurements underlying Table 1.

Table 4: Evaluator-specific results for representative LAMs using four-frame rollouts. Aggregated results appear in Table 1.

Dataset	Model	Evaluator	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
SSv2	LAPA	Gemini 3.5 Flash Lite	0.461	0.505	0.020	0.231
		Gemini 3 Flash Preview	0.567	0.522	0.035	0.286
		DeepSeek V4 Flash Vision Exp	0.498	0.433	0.010	0.219
	UniVLA	Gemini 3.5 Flash Lite	0.508	0.637	0.005	0.324
		Gemini 3 Flash Preview	0.613	0.674	0.075	0.383
		DeepSeek V4 Flash Vision Exp	0.497	0.596	0.025	0.302
	villa-X	Gemini 3.5 Flash Lite	0.538	0.611	0.015	0.330
		Gemini 3 Flash Preview	0.593	0.649	0.093	0.349
		DeepSeek V4 Flash Vision Exp	0.479	0.585	0.018	0.283
	LaWAM	Gemini 3.5 Flash Lite	0.533	0.672	0.010	0.360
		Gemini 3 Flash Preview	0.607	0.715	0.090	0.399
		DeepSeek V4 Flash Vision Exp	0.499	0.635	0.038	0.307
HRDexDB	LAPA	Gemini 3.5 Flash Lite	0.040	0.568	0.000	0.035
		Gemini 3 Flash Preview	0.387	0.620	0.005	0.258
		DeepSeek V4 Flash Vision Exp	0.250	0.639	0.005	0.201
	UniVLA	Gemini 3.5 Flash Lite	0.185	0.709	0.000	0.157
		Gemini 3 Flash Preview	0.473	0.798	0.035	0.373
		DeepSeek V4 Flash Vision Exp	0.385	0.767	0.015	0.320
	villa-X	Gemini 3.5 Flash Lite	0.098	0.528	0.000	0.080
		Gemini 3 Flash Preview	0.443	0.631	0.010	0.296
		DeepSeek V4 Flash Vision Exp	0.323	0.656	0.010	0.255
	LaWAM	Gemini 3.5 Flash Lite	0.220	0.659	0.050	0.183
		Gemini 3 Flash Preview	0.524	0.682	0.140	0.313
		DeepSeek V4 Flash Vision Exp	0.470	0.634	0.185	0.270

B Dataset Construction

B.1 Something-Something V2

We select ten action classes from Something-Something V2 (SSv2) (Goyal et al., 2017), covering upward and downward motion, lateral pushing and pulling, motion toward or away from the camera, and opening and closing. We uniformly sample eight frames over the full duration of each video. Training videos are grouped by action class and sorted by video ID within each class, with neighboring videos forming source–target pairs. During training, only a randomly selected adjacent pair from the eight sampled source frames is used. Here, adjacency refers to the sampled sequence, not to consecutive frames in the original video.

The initial evaluation pool contains 400 SSv2 videos from eight action classes, with 50 videos per class, covering upward and downward motion, lateral pushing and pulling, and opening and closing. During manual review, we verify that the action is clearly visible, the endpoint state is valid, and the target object remains in view. We also check for dominant camera motion and severe occlusion. We exclude the opening and closing classes because of their large intra-class variation, along with videos that fail the remaining quality criteria. The final SSv2 validation suite contains 200 videos across six directional actions in the image plane:

- Moving something up;
- Moving something down;
- Pushing something from left to right;
- Pushing something from right to left;
- Pulling something from left to right;
- Pulling something from right to left.

B.2 HRDexDB

HRDexDB (Lim et al., 2026) includes four embodiments: human, Allegro, Inspire DFTP, and Inspire F1. We uniformly sample 32 frames over the full duration of each training episode. The four

Table 5: Augmentation ranges for LAM pretraining.

Augmentation	Range
Random resized crop area fraction	0.800–1.000
Crop aspect ratio	0.900–1.100
Brightness factor	0.800–1.200
Contrast factor	0.800–1.200
Saturation factor	0.800–1.200

Table 6: Visual encoder configurations for the LAM variants.

Encoder	Resolution	Patch grid	Feature dim.	Layer
DINOv3/16	256×256	16×16	1024	-2
CLIP	224×224	16×16	1024	-2
MAE-H/14	224×224	16×16	1280	-2
V-JEPA 2.1 ViT-L/16_{384}	384×384	24×24	1024	-1

embodiments receive equal sampling weights, and each batch contains the same number of samples from each embodiment.

We adopt a seen-object, episode-disjoint split: objects may appear in both training and evaluation, but episodes cannot. Each human episode and all its verified robot pairings are assigned to the same split, preventing pairings of the same action instance from appearing in both training and evaluation.

The HRDexDB validation set includes only human–robot pairs with verified correspondence, successful robot execution, complete videos, and object-pose annotations for both episodes. Each pair provides two transfer directions: human-to-robot and robot-to-human. The suite contains 50 pairs, 100 directed samples, and 50 objects, with no episodes shared with the training set.

C LAM Pretraining for the Representation Design Study

This section describes the LAM variants used in Section 5.2. We pretrain entirely in feature space, keeping the visual encoder frozen while jointly training the IDM, latent bottleneck, and FDM. This stage uses no pixel decoder or RGB reconstruction loss.

C.1 Training Data and Two-View Augmentation

We use PE Video data from the Perception Encoder work and the Kuka and Fractal datasets from Open X-Embodiment (Bolya et al., 2025; O'Neill et al., 2024). From each adjacent-frame pair (I_t, I_{t+1}), we create an action view $V^a = (I_t^a, I_{t+1}^a)$ and a prediction view $V^p = (I_t^p, I_{t+1}^p)$. Augmentation parameters are sampled independently for the two views, and the same parameters are applied to both frames within each view, thereby preserving their temporal changes. The augmentation ranges are listed in Table 5.

The frozen encoder extracts features from both views. The IDM computes z_t from the action view. The FDM then predicts the next-step feature in the prediction view, $\hat{f}_{t+1}^p = \mathrm{FDM}(f_t^p, z_t)$, using f_{t+1}^p as the supervision target. The latent action must therefore support dynamics prediction across independently augmented views. As in Section 3, z_t denotes the representation after the latent bottleneck of the IDM.

C.2 Visual Encoders and Feature Processing

We use DINOv3, CLIP, MAE, and V-JEPA 2.1 (Siméoni et al., 2025; Radford et al., 2021; He et al., 2022; Mur-Labadia et al., 2026), with configurations listed in Table 6. Layer indices -1 and -2 denote the final and penultimate layers, respectively.

Table 7: IDM and FDM configurations for the LAM variants.

Configuration	IDM	FDM
Hidden dimension	1024	1024
Transformer layers	24	12
Attention heads	16	16
FFN dimension	4096	4096
Positional encoding	Fixed 3D spatiotemporal	Fixed 2D spatial
Latent action dimension	32	32 (conditioning input)
Action tokens	1	–

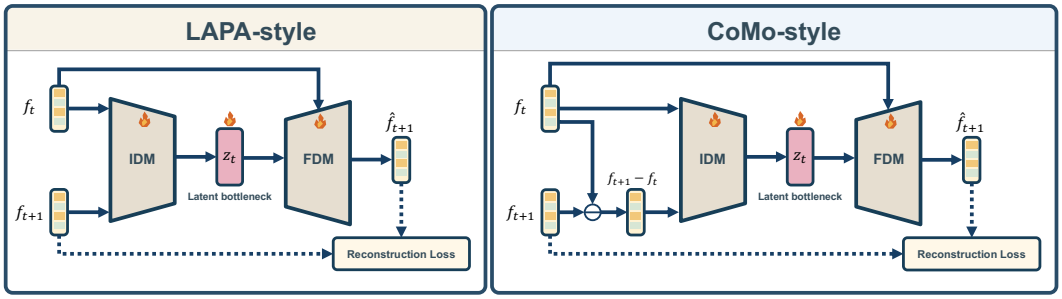


Figure 8: IDM input formulations. LAPA-style encodes features from adjacent sampled frames; CoMo-style encodes the current features and their temporal difference. Both use a latent bottleneck and an FDM trained to reconstruct future features. Augmentation-view superscripts are omitted.

We remove prefix tokens, including CLS and register tokens, and retain only the spatial patch tokens. We apply LayerNorm without learnable affine parameters along each token's feature dimension. The encoder remains frozen and in evaluation mode throughout training. The action and prediction views are combined into a tensor of shape $[2B, 2, 3, H, W]$ and processed in a single encoder call, where B is the number of original frame pairs in the local batch.

C.3 IDM and FDM Architectures

The IDM and FDM architectures follow the first stage of LaWAM (Chen et al., 2026a). Table 7 summarizes their configurations.

IDM. We project the input features to 1024 dimensions and add fixed spatiotemporal sine–cosine positional encodings. A learnable action query first attends to the visual tokens through cross-attention. The query is then concatenated with those tokens, and the resulting sequence is processed by 24 Transformer layers. An attention mask allows the query to attend to visual tokens but prevents visual tokens from attending to the query. The final query representation passes through the bottleneck to produce a single 32-dimensional latent action.

Both input variants share this architecture. The LAPA-style variant uses (f_t^a, f_{t+1}^a), whereas the CoMo-style variant uses $(f_t^a, f_{t+1}^a - f_t^a)$, explicitly providing the temporal difference along with its context (Ye et al., 2025; Yang et al., 2026). Figure 8 illustrates the two formulations.

FDM. The current prediction-view features are linearly projected and combined with fixed spatial sine–cosine positional encodings, then passed through 12 action-conditioned Transformer layers. The latent action modulates normalization in the self-attention and FFN branches through AdaLN, with gates controlling the strength of the residual connections. Zero-initialized modulation layers ensure that each block starts near an identity mapping. An output projection followed by LayerNorm produces future features with the same feature dimension as the encoder output.

Table 8: Optimization settings for LAM pretraining.

Parameter	Setting
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	0.010
Global batch size	384
Warmup	1,000 optimizer steps
Total training steps	10,000 optimizer steps
GPU hardware	$4\times$ NVIDIA H100

Table 9: Parameter counts of representative LAM variants.

Visual encoder	Total parameters (M)	Trainable parameters (M)
DINOv3	838.490	535.360
V-JEPA 2.1	840.040	535.360
CLIP	838.540	535.360
MAE	1,170.060	539.300

C.4 Latent Bottlenecks and Training Objectives

Continuous VAE bottleneck. The main IDM comparisons use a 32-dimensional Gaussian latent variable (Kingma & Welling, 2014). After LayerNorm, the query representation is mapped to a posterior mean μ and log-variance $\log \sigma^2$. Training uses the reparameterization $z = \mu + \sigma\epsilon$, while evaluation uses the posterior mean μ. The log-variance is clamped to $[-10, 10]$, and the KL regularization weight is set to 5×10^{-5}.

Discrete VQ-VAE bottleneck. The discrete variants use 32-dimensional codewords with codebook sizes of 16, 32, and 64 (Van Den Oord et al., 2017). Gradients pass through a straight-through estimator. The codebook and commitment loss weights are 1.0 and 0.25, respectively.

The feature prediction loss is $\mathcal{L}_{\text{rec}} = \text{SmoothL1}(\hat{f}^p_{t+1}, \text{stopgrad}(f^p_{t+1}))$, with a transition threshold of 0.1. The total loss adds the weighted bottleneck term to $\mathcal{L}_{\text{rec}}$: KL regularization for the VAE, or codebook and commitment losses for the VQ-VAE, using the weights specified above.

C.5 Optimization and Parameter Counts

Table 8 lists the shared hardware and optimization settings for LAM pretraining.

Table 9 reports representative model sizes. Total parameter counts include the frozen visual encoder, whereas trainable counts cover the IDM, latent bottleneck, and FDM. Both exclude the pixel decoder used for transfer evaluation.

D Pixel Decoder Architectures and Training

This section describes the pixel decoder used in the unified transfer protocol. It predicts the next frame from a latent action and a reference image.

D.1 Training Procedure

We uniformly sample eight frames from each SSv2 training video and 32 from each HRDexDB training video (Appendix B), then randomly select an adjacent-frame pair (I_t, I_{t+1}) from the sampled sequence. The frozen visual encoder and IDM extract $z_t = \text{IDM}(f_t, f_{t+1})$, where $f_t = \text{Encoder}(I_t)$ and z_t is the output of the latent bottleneck.

Table 10: Pixel decoder configurations and approximate trainable parameter counts.

LAM	Output	Patch size	Action dim.	Parameters
LaWAM	256^2	16	32	≈ 152.800M
UniVLA	224^2	14	128	≈ 152.500M
LAPA	224^2	32	1024	≈ 158.500M
villa-X	224^2	14	64	≈ 152.400M

The pixel decoder takes the current ground-truth frame as its reference and predicts $\hat{I}_{t+1} = G(I_t, z_t)$. The FDM is not used during decoder training or generation.

The decoder is supervised with I_{t+1} as the RGB target. Only the decoder parameters are updated; the representation extractor remains frozen. During transfer evaluation, actions are extracted from the source video. Generation starts from the target initial frame and uses the previous prediction as the reference at each subsequent step.

D.2 Pixel Decoder Architecture

The pixel decoder consists of an action branch and a reference-image branch, followed by a Transformer that fuses their tokens.

Action branch. Latent actions have shape $B \times Q \times D_a$, where B is the batch size, Q is the number of action tokens, and D_a is their feature dimension. The VAE and VQ-VAE variants in our representation design study use $Q = 1$ and $D_a = 32$; existing LAMs retain their model-specific token counts and dimensions. A linear layer projects the action tokens to 1024 dimensions, after which fixed one-dimensional sine–cosine positional encodings and a learnable action-type embedding are added.

Reference-image branch. The reference image is resized to the output resolution and divided into non-overlapping $P \times P$ RGB patches. Each patch is flattened into a $3P^2$-dimensional vector and linearly projected to 1024 dimensions. Fixed two-dimensional spatial sine–cosine positional encodings and a learnable reference-type embedding are then added.

Fusion and output. Action and reference tokens are concatenated along the sequence dimension and processed by 12 Transformer encoder layers. Only the reference tokens are used for RGB prediction. LayerNorm followed by a linear projection maps each output token to $3P^2$ values. These values are rearranged into an image and passed through a sigmoid to produce RGB values in $[0, 1]$.

The Transformer uses 16 attention heads, an FFN dimension of 4096, GELU activations, and pre-norm normalization. We adapt the decoder to each LAM's output resolution, patch size, and action dimension. Table 10 lists the trainable parameter counts, excluding frozen LAM components.

D.3 Training Objective

The pixel decoder is trained with a pixel-level L1 loss, $\mathcal{L}_{\text{pixel}} = \text{mean}\,|\hat{I}_{t+1} - I_{t+1}|$, averaged over the batch, spatial positions, and RGB channels.

D.4 Optimization

All pixel decoders use the hardware and optimization settings listed in Table 11.

E Semantic Structure of Latent Action Representations

We compare LaWAM, UniVLA, villa-X, and LAPA on the SSv2 validation set to examine the semantic structure of their latent actions and its relationship to transfer performance. We uniformly sample eight frames from each video and use each model's IDM to extract seven latent actions from

Table 11: Optimization settings for pixel decoder training.

Parameter	Setting
GPU hardware	$4\times$ NVIDIA H100
Per-GPU batch size	64
Global batch size	256
Optimizer	AdamW
Learning rate	3×10^{-4}
Weight decay	0.010
Total training steps	45,000
Warmup steps	6,000
Post-warmup schedule	Cosine decay
Training precision	BF16
Gradient clipping threshold	1.000

consecutive sampled frames. For each latent action, we average over the action tokens and apply L2 normalization to the resulting vector. The seven normalized vectors are then concatenated in temporal order to form a video-level action representation.

We reduce the dimensionality of each model's representations independently. We first apply principal component analysis (PCA), setting the output dimension to one quarter of the original feature dimension, rounded up. If this dimension reaches or exceeds the maximum possible rank of the centered data, we retain the centered features directly. We then project the representations to two dimensions using t-SNE with Euclidean distance, PCA initialization, an automatic learning rate, a perplexity of 30, and a maximum of 2,000 iterations. The random seed is fixed at 42. Each point represents one video, colors indicate action classes, and crosses mark the coordinate-wise median of each class in the two-dimensional projection. Because the projections are fitted independently, we compare patterns of class overlap and separation within each plot rather than coordinates or distances across plots.

In the joint projection of all six action classes, LaWAM exhibits clearer class structure, with different actions tending to occupy distinct regions (Figure 9). Some classes still overlap. UniVLA and villa-X show local groupings for certain classes but more mixing overall, while the classes in LAPA overlap broadly. To examine directional information, we also consider three pairs of opposite actions: moving up versus down, pushing left to right versus right to left, and pulling left to right versus right to left. We fit the same dimensionality-reduction pipeline independently for each model and action pair. LaWAM shows clearer directional separation across all three pairs (Figure 10). UniVLA partly separates upward and downward motion, but its other pairs overlap substantially. Opposite actions also remain mixed in villa-X and LAPA.

These visualizations suggest that the latent actions of LaWAM encode distinctions between action classes and opposite directions more clearly. This pattern is consistent with its highest mean Transfer Score on SSv2 validation among the four models (Table 1). Clearer encoding of action differences may help the decoder reproduce the intended action in a new context. The observed structure thus offers a representation-level account of the strong transfer performance of LaWAM on SSv2.

F Complete Representation Design Results

F.1 Visual Features and Learned Latent Actions

Table 12 reports the per-evaluator measurements underlying Table 3.

F.2 Encoder and IDM Results

Table 13 reports the encoder and IDM comparison described in Section 5.2. Tables 14 and 15 provide the per-evaluator results.

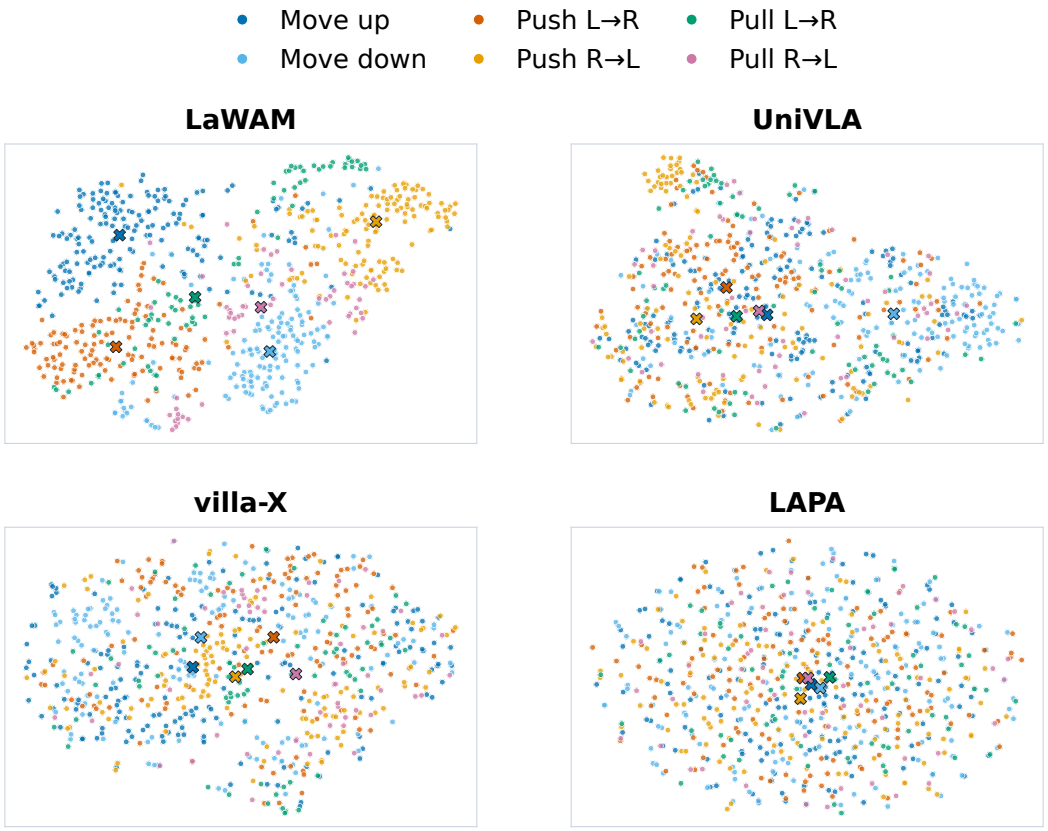


Figure 9: Video-level latent action representations for six action classes on SSv2 validation. Each model's t-SNE projection is fitted independently. Points represent videos, colors indicate action classes, and crosses mark coordinate-wise class medians. LaWAM shows clearer class structure, whereas the other models exhibit more overlap.

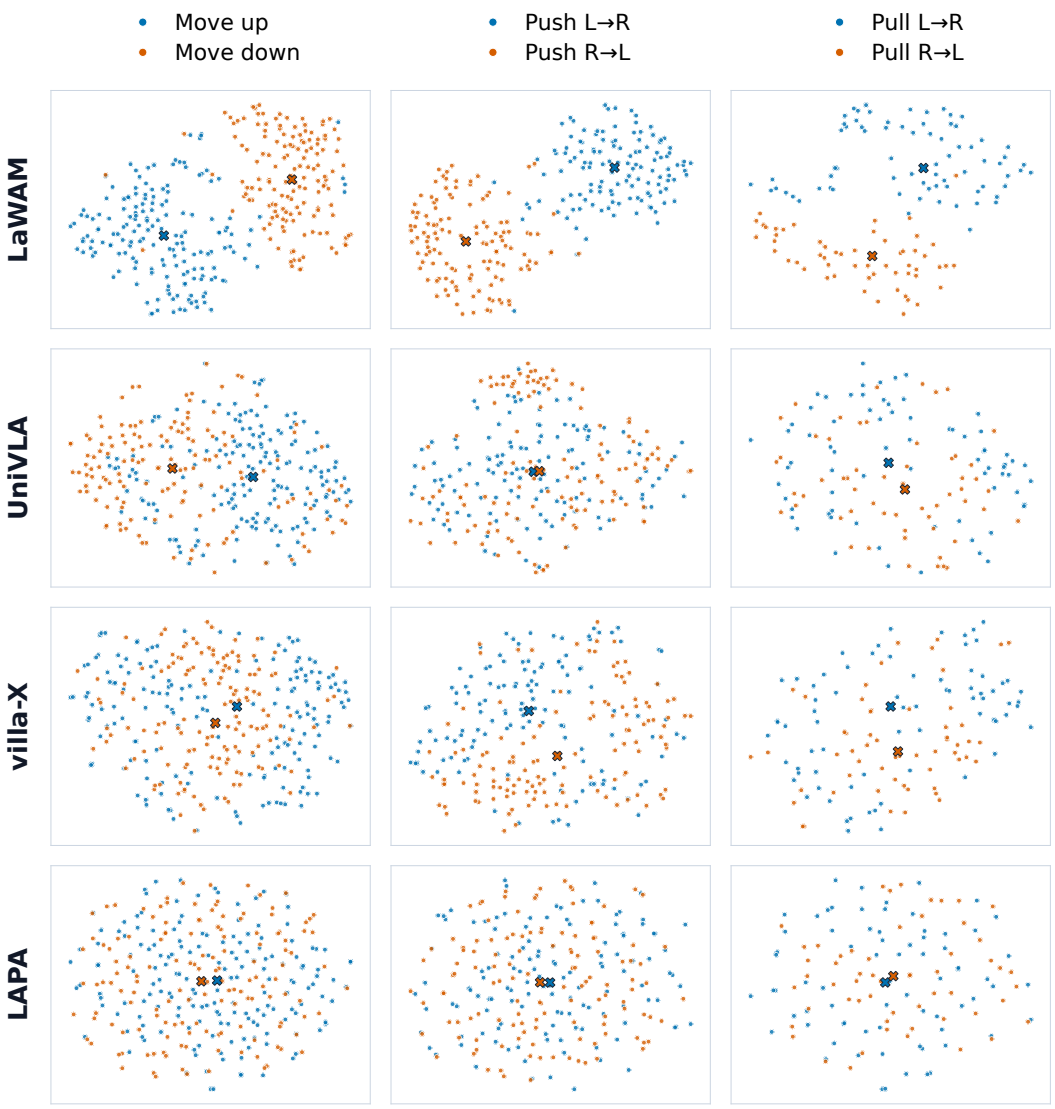


Figure 10: Video-level latent action representations for opposite actions on SSv2 validation. Rows correspond to LaWAM, UniVLA, villa-X, and LAPA; columns correspond to opposite directions of moving, pushing, and pulling. Each projection is fitted independently on the corresponding action pair. Colors distinguish the two directions, and crosses mark the coordinate-wise class medians. LaWAM exhibits clearer directional separation across all three pairs.

Table 12: Per-evaluator comparison of general visual feature differences and LAPA-style latent actions on SSv2 validation using four-frame rollouts.

Representation	Evaluator	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
DINOv2	Gemini 3.5 Flash Lite	0.663	0.652	0.383	0.268
	Gemini 3 Flash Preview	0.610	0.673	0.715	0.132
	DeepSeek V4 Flash Vision Exp	0.530	0.659	0.510	0.169
DINOv3	Gemini 3.5 Flash Lite	0.614	0.675	0.248	0.313
	Gemini 3 Flash Preview	0.598	0.682	0.675	0.140
	DeepSeek V4 Flash Vision Exp	0.603	0.650	0.383	0.231
LAPA-DINOv2	Gemini 3.5 Flash Lite	0.553	0.681	0.005	0.374
	Gemini 3 Flash Preview	0.613	0.721	0.093	0.403
	DeepSeek V4 Flash Vision Exp	0.524	0.629	0.038	0.317
LAPA-DINOv3	Gemini 3.5 Flash Lite	0.601	0.664	0.005	0.403
	Gemini 3 Flash Preview	0.616	0.695	0.098	0.390
	DeepSeek V4 Flash Vision Exp	0.540	0.612	0.033	0.329

Table 13: Visual encoder and IDM results on SSv2 and HRDexDB validation. Metrics are reported as the mean ± standard deviation across three evaluator models. Bold indicates the highest mean Transfer Score for each dataset.

Encoder	IDM	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
SSv2 validation					
DINOv3	LAPA-style	0.543 ± 0.045	0.641 ± 0.037	0.039 ± 0.044	0.337 ± 0.034
	CoMo-style	0.579 ± 0.054	0.656 ± 0.023	0.061 ± 0.050	$\mathbf{0.361 \pm 0.045}$
V-JEPA 2.1	LAPA-style	0.549 ± 0.038	0.654 ± 0.033	0.058 ± 0.048	0.333 ± 0.022
	CoMo-style	0.547 ± 0.045	0.678 ± 0.031	0.043 ± 0.044	0.354 ± 0.033
CLIP	LAPA-style	0.533 ± 0.043	0.630 ± 0.040	0.036 ± 0.036	0.321 ± 0.033
	CoMo-style	0.554 ± 0.046	0.625 ± 0.031	0.049 ± 0.051	0.331 ± 0.026
MAE	LAPA-style	0.520 ± 0.050	0.586 ± 0.042	0.019 ± 0.024	0.294 ± 0.043
	CoMo-style	0.508 ± 0.055	0.585 ± 0.042	0.023 ± 0.020	0.291 ± 0.046
HRDexDB validation					
DINOv3	LAPA-style	0.359 ± 0.110	0.643 ± 0.027	0.098 ± 0.070	0.246 ± 0.052
	CoMo-style	0.429 ± 0.119	0.575 ± 0.034	0.187 ± 0.090	0.217 ± 0.035
V-JEPA 2.1	LAPA-style	0.323 ± 0.149	0.543 ± 0.030	0.143 ± 0.080	0.174 ± 0.068
	CoMo-style	0.424 ± 0.111	0.657 ± 0.037	0.030 ± 0.023	$\mathbf{0.296 \pm 0.064}$
CLIP	LAPA-style	0.291 ± 0.148	0.567 ± 0.027	0.143 ± 0.091	0.161 ± 0.071
	CoMo-style	0.439 ± 0.141	0.630 ± 0.010	0.203 ± 0.126	0.236 ± 0.040
MAE	LAPA-style	0.271 ± 0.131	0.705 ± 0.031	0.013 ± 0.010	0.229 ± 0.097
	CoMo-style	0.248 ± 0.141	0.696 ± 0.033	0.005 ± 0.007	0.209 ± 0.108

F.3 Bottleneck Results

Table 16 reports the bottleneck comparison on SSv2 validation. All configurations use LAPA-style IDMs. V-JEPA 2.1 uses ViT-L/16_{384}, and DINOv3 uses the large backbone at resolution 256. Table 17 provides the per-evaluator results for all configurations.

G Complete Rollout-Length Results

Table 18 reports the aggregate results underlying Figure 5; Table 19 provides all per-evaluator scores. Each prefix contains $F \in \{2, 4, 6, 8\}$ frames, including the initial target frame, sampled as described in Appendix A.2. Aggregate results are the mean ± population standard deviation across the three evaluators. Figure 5 uses the same means with error bars of ± 0.5 standard deviations. Transfer Scores are aggregated from sample-level scores as described in Appendix A.6.

Table 14: Per-evaluator encoder and IDM results on SSv2 validation.

Encoder	IDM	Evaluator	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
DINOv3	LAPA-style	Gemini 3.5 Flash Lite	0.529	0.648	0.000	0.349
		Gemini 3 Flash Preview	0.604	0.682	0.100	0.371
		DeepSeek V4 Flash Vision Exp	0.496	0.594	0.018	0.290
	CoMo-style	Gemini 3.5 Flash Lite	0.606	0.662	0.000	0.404
		Gemini 3 Flash Preview	0.626	0.681	0.123	0.380
		DeepSeek V4 Flash Vision Exp	0.504	0.625	0.060	0.299
V-JEPA 2.1	LAPA-style	Gemini 3.5 Flash Lite	0.525	0.659	0.015	0.328
		Gemini 3 Flash Preview	0.602	0.691	0.125	0.363
		DeepSeek V4 Flash Vision Exp	0.519	0.611	0.033	0.309
	CoMo-style	Gemini 3.5 Flash Lite	0.538	0.672	0.010	0.355
		Gemini 3 Flash Preview	0.606	0.719	0.105	0.394
		DeepSeek V4 Flash Vision Exp	0.498	0.644	0.015	0.314
CLIP	LAPA-style	Gemini 3.5 Flash Lite	0.520	0.634	0.000	0.322
		Gemini 3 Flash Preview	0.591	0.677	0.085	0.360
		DeepSeek V4 Flash Vision Exp	0.489	0.579	0.023	0.279
	CoMo-style	Gemini 3.5 Flash Lite	0.564	0.625	0.005	0.346
		Gemini 3 Flash Preview	0.604	0.663	0.120	0.352
		DeepSeek V4 Flash Vision Exp	0.494	0.588	0.023	0.294
MAE	LAPA-style	Gemini 3.5 Flash Lite	0.528	0.596	0.005	0.303
		Gemini 3 Flash Preview	0.577	0.632	0.053	0.341
		DeepSeek V4 Flash Vision Exp	0.456	0.530	0.000	0.237
	CoMo-style	Gemini 3.5 Flash Lite	0.484	0.591	0.005	0.281
		Gemini 3 Flash Preview	0.584	0.633	0.050	0.351
		DeepSeek V4 Flash Vision Exp	0.456	0.531	0.013	0.240

Table 15: Per-evaluator encoder and IDM results on HRDexDB validation.

Encoder	IDM	Evaluator	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
DINOv3	LAPA-style	Gemini 3.5 Flash Lite	0.213	0.605	0.000	0.175
		Gemini 3 Flash Preview	0.478	0.669	0.140	0.300
		DeepSeek V4 Flash Vision Exp	0.388	0.655	0.155	0.263
	CoMo-style	Gemini 3.5 Flash Lite	0.263	0.575	0.060	0.168
		Gemini 3 Flash Preview	0.531	0.616	0.260	0.247
		DeepSeek V4 Flash Vision Exp	0.493	0.533	0.240	0.236
V-JEPA 2.1	LAPA-style	Gemini 3.5 Flash Lite	0.118	0.508	0.030	0.080
		Gemini 3 Flash Preview	0.467	0.582	0.205	0.239
		DeepSeek V4 Flash Vision Exp	0.385	0.540	0.195	0.204
	CoMo-style	Gemini 3.5 Flash Lite	0.275	0.605	0.000	0.210
		Gemini 3 Flash Preview	0.540	0.691	0.035	0.366
		DeepSeek V4 Flash Vision Exp	0.457	0.674	0.055	0.313
CLIP	LAPA-style	Gemini 3.5 Flash Lite	0.088	0.535	0.015	0.062
		Gemini 3 Flash Preview	0.435	0.601	0.210	0.224
		DeepSeek V4 Flash Vision Exp	0.350	0.566	0.205	0.197
	CoMo-style	Gemini 3.5 Flash Lite	0.240	0.617	0.025	0.180
		Gemini 3 Flash Preview	0.558	0.636	0.300	0.266
		DeepSeek V4 Flash Vision Exp	0.518	0.638	0.285	0.263
MAE	LAPA-style	Gemini 3.5 Flash Lite	0.100	0.662	0.000	0.096
		Gemini 3 Flash Preview	0.418	0.733	0.025	0.324
		DeepSeek V4 Flash Vision Exp	0.295	0.720	0.015	0.268
	CoMo-style	Gemini 3.5 Flash Lite	0.063	0.649	0.000	0.059
		Gemini 3 Flash Preview	0.405	0.720	0.015	0.310
		DeepSeek V4 Flash Vision Exp	0.278	0.719	0.000	0.258

H Correlation Between Human and VLM Scores

To assess the agreement between automated VLM scores and human judgments, we select 10 source–target video pairs from SSv2 validation and 10 from HRDexDB validation, yielding 80 action-transfer outputs generated by LaWAM, UniVLA, villa-X, and LAPA. Three human raters evaluate these outputs without knowing which model produced each one. For each output, we separately average the Transfer Scores assigned by the three human raters and by the three VLM evaluators, then compute the correlations between the two sets of mean scores across all 80 outputs.

Table 16: Bottleneck results on SSv2 validation, reported as the mean ± standard deviation across three evaluator models. K denotes the VQ-VAE codebook size. Bold highlights the best discrete Transfer Score for each encoder.

Encoder	Bottleneck	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
V-JEPA 2.1	VAE	0.549 ± 0.038	0.654 ± 0.033	0.058 ± 0.048	0.333 ± 0.022
	VQ-VAE ($K = 16$)	0.517 ± 0.045	0.617 ± 0.044	0.026 ± 0.020	$\mathbf{0.310 \pm 0.043}$
	VQ-VAE ($K = 32$)	0.513 ± 0.052	0.590 ± 0.043	0.030 ± 0.029	0.290 ± 0.041
	VQ-VAE ($K = 64$)	0.511 ± 0.046	0.629 ± 0.038	0.028 ± 0.026	0.306 ± 0.044
DINOv3	VAE	0.543 ± 0.045	0.641 ± 0.037	0.039 ± 0.044	0.337 ± 0.034
	VQ-VAE ($K = 16$)	0.479 ± 0.060	0.554 ± 0.044	0.029 ± 0.029	0.255 ± 0.047
	VQ-VAE ($K = 32$)	0.524 ± 0.047	0.563 ± 0.045	0.022 ± 0.022	$\mathbf{0.286 \pm 0.042}$
	VQ-VAE ($K = 64$)	0.509 ± 0.042	0.567 ± 0.046	0.018 ± 0.011	0.281 ± 0.044

Table 17: Per-evaluator VAE and VQ-VAE bottleneck results on SSv2 validation.

Configuration	Evaluator	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
DINOv3 VAE	Gemini 3.5 Flash Lite	0.529	0.648	0.000	0.349
	Gemini 3 Flash Preview	0.604	0.682	0.100	0.371
	DeepSeek V4 Flash Vision Exp	0.496	0.594	0.018	0.290
DINOv3 VQ-VAE ($K = 16$)	Gemini 3.5 Flash Lite	0.435	0.556	0.010	0.234
	Gemini 3 Flash Preview	0.564	0.607	0.070	0.319
	DeepSeek V4 Flash Vision Exp	0.438	0.500	0.008	0.211
DINOv3 VQ-VAE ($K = 32$)	Gemini 3.5 Flash Lite	0.528	0.562	0.005	0.283
	Gemini 3 Flash Preview	0.580	0.619	0.053	0.338
	DeepSeek V4 Flash Vision Exp	0.465	0.509	0.008	0.237
DINOv3 VQ-VAE ($K = 64$)	Gemini 3.5 Flash Lite	0.483	0.569	0.005	0.263
	Gemini 3 Flash Preview	0.568	0.622	0.033	0.342
	DeepSeek V4 Flash Vision Exp	0.475	0.510	0.018	0.239
V-JEPA 2.1 VAE	Gemini 3.5 Flash Lite	0.525	0.659	0.015	0.328
	Gemini 3 Flash Preview	0.602	0.691	0.125	0.363
	DeepSeek V4 Flash Vision Exp	0.519	0.611	0.033	0.309
V-JEPA 2.1 VQ-VAE ($K = 16$)	Gemini 3.5 Flash Lite	0.491	0.627	0.005	0.307
	Gemini 3 Flash Preview	0.581	0.665	0.053	0.364
	DeepSeek V4 Flash Vision Exp	0.479	0.558	0.020	0.259
V-JEPA 2.1 VQ-VAE ($K = 32$)	Gemini 3.5 Flash Lite	0.476	0.596	0.015	0.269
	Gemini 3 Flash Preview	0.587	0.639	0.070	0.348
	DeepSeek V4 Flash Vision Exp	0.477	0.534	0.005	0.254
V-JEPA 2.1 VQ-VAE ($K = 64$)	Gemini 3.5 Flash Lite	0.513	0.630	0.000	0.317
	Gemini 3 Flash Preview	0.566	0.674	0.063	0.354
	DeepSeek V4 Flash Vision Exp	0.454	0.581	0.023	0.248

The Spearman rank correlation is $\rho = 0.749$, and the Pearson correlation is $r = 0.778$. These results indicate strong rank and linear correlations between the VLM ensemble scores and human judgments, supporting their use as an automated proxy for action-transfer quality.

I Additional Qualitative Results

Figures 11 and 12 each present ten transfer examples with eight frames per sequence.

I.1 Camera Motion and Cross-View Transfer

Figure 13 extends the analysis in Sections 5.4.2 and 5.4.3. In the SSv2 examples, generated rollouts often combine the intended object motion with global displacement and increasing blur, disrupting the target object and scene. In the RH20T examples, some rollouts preserve the overall target-view layout, while others develop ghosting, corrupted textures, or a loss of scene structure over time. Across these source–target pairs, the examples illustrate the difficulty of disentangling action from camera and viewpoint variation.

Table 18: SSv2 transfer across rollout lengths, reported as the mean ± one population standard deviation across three evaluators.

Model	F	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
LaWAM	2	0.4390 ± 0.0894	0.8210 ± 0.0407	0.0117 ± 0.0092	0.3530 ± 0.0869
LaWAM	4	0.5460 ± 0.0451	0.6738 ± 0.0326	0.0458 ± 0.0332	0.3552 ± 0.0373
LaWAM	6	0.6294 ± 0.0301	0.5538 ± 0.0252	0.0925 ± 0.0776	0.3232 ± 0.0323
LaWAM	8	0.6309 ± 0.0206	0.4714 ± 0.0238	0.1258 ± 0.1184	0.2647 ± 0.0407
UniVLA	2	0.4581 ± 0.0650	0.7854 ± 0.0372	0.0042 ± 0.0042	0.3506 ± 0.0671
UniVLA	4	0.5392 ± 0.0524	0.6357 ± 0.0320	0.0350 ± 0.0294	0.3361 ± 0.0343
UniVLA	6	0.6062 ± 0.0212	0.5186 ± 0.0294	0.0708 ± 0.0707	0.2990 ± 0.0272
UniVLA	8	0.5731 ± 0.0211	0.4369 ± 0.0258	0.0992 ± 0.0992	0.2353 ± 0.0228
villa-X	2	0.4302 ± 0.0753	0.7702 ± 0.0345	0.0108 ± 0.0153	0.3223 ± 0.0732
villa-X	4	0.5365 ± 0.0466	0.6153 ± 0.0262	0.0417 ± 0.0360	0.3209 ± 0.0276
villa-X	6	0.5645 ± 0.0488	0.4785 ± 0.0253	0.0725 ± 0.0731	0.2582 ± 0.0254
villa-X	8	0.5482 ± 0.0306	0.3959 ± 0.0212	0.1225 ± 0.1123	0.2000 ± 0.0169
LAPA	2	0.4622 ± 0.0605	0.6174 ± 0.0481	0.0117 ± 0.0103	0.2850 ± 0.0494
LAPA	4	0.5088 ± 0.0438	0.4867 ± 0.0387	0.0217 ± 0.0103	0.2451 ± 0.0293
LAPA	6	0.4888 ± 0.0540	0.3973 ± 0.0363	0.0442 ± 0.0361	0.1896 ± 0.0214
LAPA	8	0.4750 ± 0.0728	0.3398 ± 0.0341	0.0625 ± 0.0586	0.1541 ± 0.0180

J Do the Generated Dynamics Come from the Pixel Decoder?

We test whether latent actions control the generated motion or whether the pixel decoder simply follows its learned dynamics. Using LaWAM and its pixel decoder, we compare standard transfer with static and reverse conditions while keeping the model weights fixed. The decoder was not trained on specially constructed static or reversed sequences.

Standard transfer uses latent actions extracted from adjacent sampled source frames and starts from the initial frame of the target video. For the static condition, we feed two copies of the same source image into the LAM to obtain a latent action representing zero visual change, then apply it to the target initial state. For the reverse condition, we extract a new latent-action sequence from the reversed source video and initialize generation from the final frame of the target video.

In Figure 14, target subjects remain largely stationary under static conditioning, while reverse conditioning produces motion trends opposite to those observed in standard transfer. Some sequences exhibit blur or deformation, yet their motion varies with the latent-action condition. These results indicate that latent actions provide a meaningful control signal rather than the decoder merely continuing a fixed motion pattern from the target image.

Table 19: Per-evaluator SSv2 results across rollout lengths for LaWAM, UniVLA, villa-X, and LAPA.

Model	F	Evaluator	AEF ↑	TCP ↑	SCL ↓	Transfer Score ↑
LaWAM	2	Gemini 3.5 Flash Lite	0.3200	0.7920	0.0000	0.2451
		Gemini 3 Flash Preview	0.5356	0.8785	0.0225	0.4578
		DeepSeek V4 Flash Vision Exp	0.4613	0.7925	0.0125	0.3561
	4	Gemini 3.5 Flash Lite	0.5325	0.6717	0.0100	0.3598
		Gemini 3 Flash Preview	0.6068	0.7147	0.0900	0.3985
		DeepSeek V4 Flash Vision Exp	0.4988	0.6349	0.0375	0.3074
	6	Gemini 3.5 Flash Lite	0.6525	0.5805	0.1975	0.3090
		Gemini 3 Flash Preview	0.6488	0.5608	0.0125	0.3679
		DeepSeek V4 Flash Vision Exp	0.5868	0.5201	0.0675	0.2927
	8	Gemini 3.5 Flash Lite	0.6475	0.4809	0.0150	0.3208
		Gemini 3 Flash Preview	0.6433	0.4945	0.2900	0.2255
		DeepSeek V4 Flash Vision Exp	0.6018	0.4387	0.0725	0.2479
UniVLA	2	Gemini 3.5 Flash Lite	0.3688	0.7588	0.0000	0.2663
		Gemini 3 Flash Preview	0.5217	0.8380	0.0100	0.4304
		DeepSeek V4 Flash Vision Exp	0.4838	0.7595	0.0025	0.3551
	4	Gemini 3.5 Flash Lite	0.5075	0.6369	0.0050	0.3238
		Gemini 3 Flash Preview	0.6130	0.6743	0.0750	0.3829
		DeepSeek V4 Flash Vision Exp	0.4970	0.5959	0.0250	0.3015
	6	Gemini 3.5 Flash Lite	0.6200	0.5319	0.0100	0.3373
		Gemini 3 Flash Preview	0.6223	0.5461	0.1700	0.2831
		DeepSeek V4 Flash Vision Exp	0.5763	0.4779	0.0325	0.2767
	8	Gemini 3.5 Flash Lite	0.5600	0.4468	0.0100	0.2674
		Gemini 3 Flash Preview	0.6029	0.4623	0.2375	0.2216
		DeepSeek V4 Flash Vision Exp	0.5563	0.4016	0.0500	0.2169
villa-X	2	Gemini 3.5 Flash Lite	0.3400	0.7465	0.0000	0.2339
		Gemini 3 Flash Preview	0.5244	0.8190	0.0325	0.4132
		DeepSeek V4 Flash Vision Exp	0.4263	0.7450	0.0000	0.3199
	4	Gemini 3.5 Flash Lite	0.5380	0.6114	0.0150	0.3303
		Gemini 3 Flash Preview	0.5928	0.6491	0.0925	0.3491
		DeepSeek V4 Flash Vision Exp	0.4788	0.5854	0.0175	0.2834
	6	Gemini 3.5 Flash Lite	0.5663	0.4810	0.0100	0.2818
		Gemini 3 Flash Preview	0.6233	0.5082	0.1750	0.2699
		DeepSeek V4 Flash Vision Exp	0.5038	0.4463	0.0325	0.2229
	8	Gemini 3.5 Flash Lite	0.5225	0.3995	0.0150	0.2235
		Gemini 3 Flash Preview	0.5912	0.4198	0.2775	0.1847
		DeepSeek V4 Flash Vision Exp	0.5310	0.3683	0.0750	0.1918
LAPA	2	Gemini 3.5 Flash Lite	0.3950	0.6325	0.0000	0.2522
		Gemini 3 Flash Preview	0.5417	0.6673	0.0250	0.3548
		DeepSeek V4 Flash Vision Exp	0.4500	0.5525	0.0100	0.2481
	4	Gemini 3.5 Flash Lite	0.4613	0.5050	0.0200	0.2306
		Gemini 3 Flash Preview	0.5670	0.5222	0.0350	0.2860
		DeepSeek V4 Flash Vision Exp	0.4980	0.4329	0.0100	0.2188
	6	Gemini 3.5 Flash Lite	0.4263	0.4098	0.0150	0.1793
		Gemini 3 Flash Preview	0.5580	0.4342	0.0950	0.2194
		DeepSeek V4 Flash Vision Exp	0.4820	0.3479	0.0225	0.1700
	8	Gemini 3.5 Flash Lite	0.3738	0.3502	0.0150	0.1333
		Gemini 3 Flash Preview	0.5418	0.3754	0.1450	0.1772
		DeepSeek V4 Flash Vision Exp	0.5093	0.2938	0.0275	0.1519

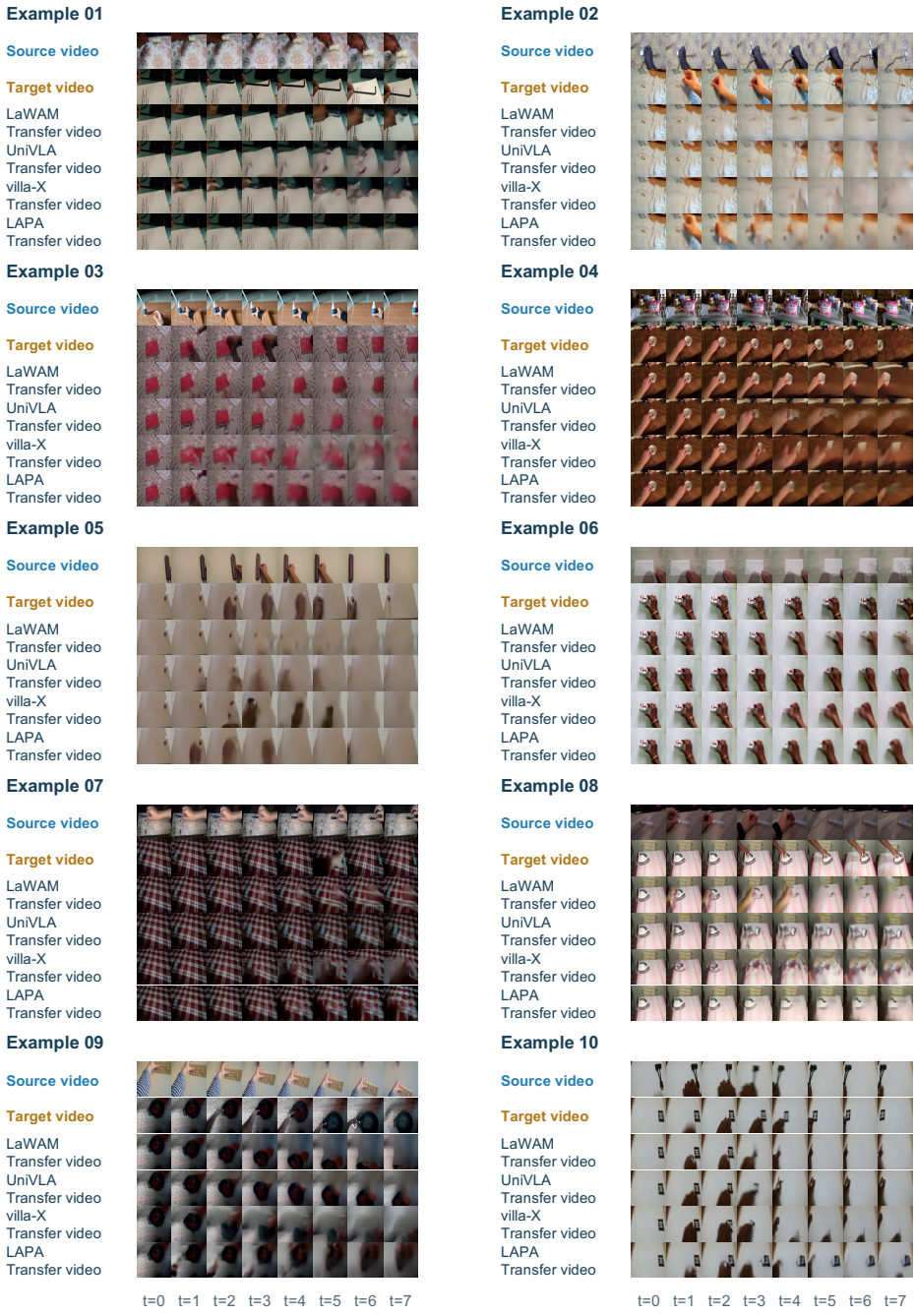


Figure 11: Additional transfer examples on SSv2. Each example shows, from top to bottom, the source video, target video, and transferred videos from LaWAM, UniVLA, villa-X, and LAPA. Columns follow temporal order from $t = 0$ to $t = 7$.

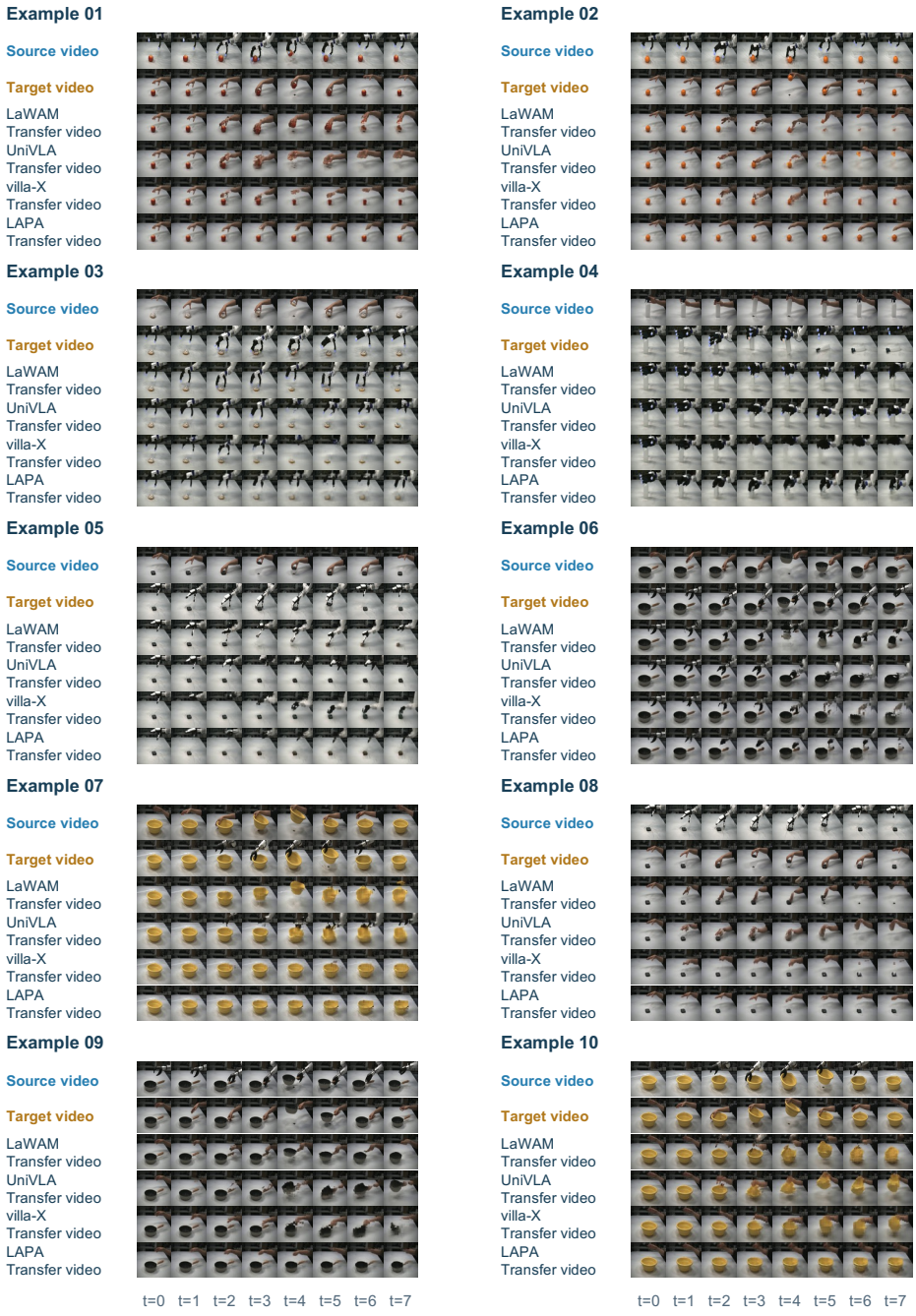


Figure 12: Additional cross-embodiment transfer examples on HRDexDB. Each example shows, from top to bottom, the source video, target video, and transferred videos from LaWAM, UniVLA, villa-X, and LAPA. Columns follow temporal order from $t = 0$ to $t = 7$.



Figure 13: Additional qualitative examples for camera-motion and cross-view transfer. Left: five SSv2 examples with substantial source-camera motion. Right: five RH20T cross-view examples. Within each example, rows show the source, target, and transfer videos, and columns follow temporal order from $t = 0$ to $t = 7$.

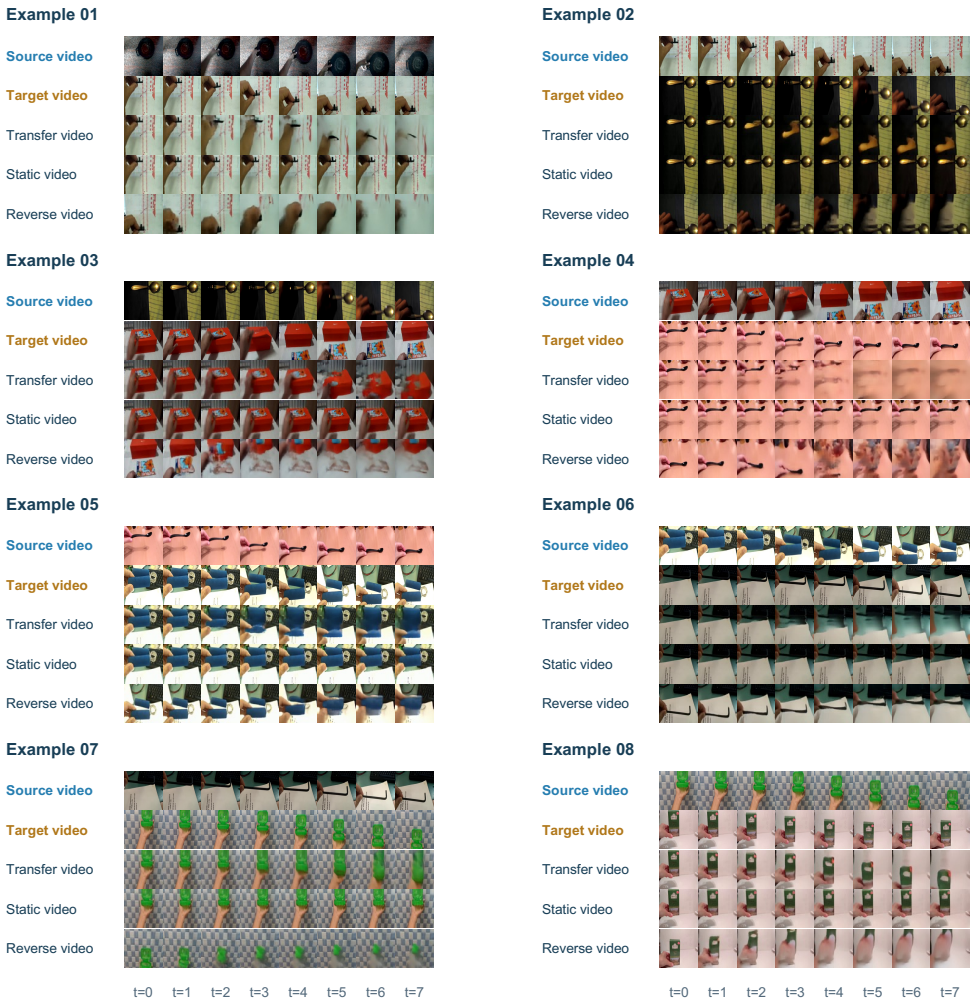


Figure 14: Responses of LaWAM to static and reverse latent actions. Each example shows, from top to bottom, the source video, target video, standard transfer, static generation, and reverse generation. Static actions are extracted from repeated source frames. Reverse actions are extracted from the reversed source video, with generation initialized from the final frame of the target video. Columns follow generation order.