LEARNING ACTION-EFFECT LATENTS FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) policies are typically trained from embodiment-specific motor commands, creating a mismatch between rich multimodal representations and the low-level signals used for action supervision. We introduce **AEL (Action-Effect Latents)**, a two-stage framework that augments action representation learning with the semantic visual change induced by executed actions. In Stage I, an action encoder represents an executed action chunk in the context of the current scene. The latent is trained to preserve the motor trajectory through action reconstruction while predicting the corresponding change in semantic visual features over the same execution horizon. This grounds the representation in both executable control and its observed scene-conditioned effect. In Stage II, the learned Action-Effect Latent becomes the prediction target of a VLA policy, which infers it from the current visual observations, language instruction, and robot state and decodes it into continuous actions. Future observations are used only as supervision during Stage I and are not required at deployment. Separating representation learning from policy learning also allows failed or suboptimal interactions to provide action–effect supervision without treating them as imitation targets. Experiments across LIBERO, CALVIN, and real-world single-arm and bimanual manipulation demonstrate consistent improvements across different task distributions and embodiments.

1 INTRODUCTION

Vision-Language-Action (VLA) models have advanced general-purpose robot manipulation by combining pretrained vision-language representations with robot control (Driess et al., 2023; Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024). This progress has been supported by increasingly diverse robot datasets and generalist policies spanning multiple tasks and embodiments (Open X-Embodiment Collaboration, 2024; Walke et al., 2023; Khazatsky et al., 2024; Ghosh et al., 2024). Most VLA policies, however, are still trained primarily from demonstrated motor commands (Brohan et al., 2022; Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024). Although directly executable, these targets are embodiment-specific and encode low-level control, whereas visual and language inputs contain richer information about objects, spatial relations, and task goals. This creates a mismatch between the multimodal representations used for perception and the signals used for action supervision.

Latent action models introduce an intermediate representation between multimodal perception and low-level control (Ye et al., 2025; Bu et al., 2025; Routray et al., 2026). Such a representation should preserve not only executable motor information but also what the action accomplishes in the current scene. This is important because similar motions can produce different outcomes under different contacts or object configurations. Action reconstruction ensures that the latent remains decodable into the executed trajectory, but does not explicitly require it to retain information about the resulting environmental change. We therefore use the semantic visual change induced by an executed action as additional supervision. The observed transition is naturally available from robot trajectories without extra annotation and provides scene-conditioned information about the action's effect. Unlike approaches that use predicted futures for policy learning, planning, or intermediate predictive representations (Wu et al., 2024; Yang et al., 2024; Routray et al., 2026), AEL uses future observations only during Stage I representation learning and does not require them at deployment.

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

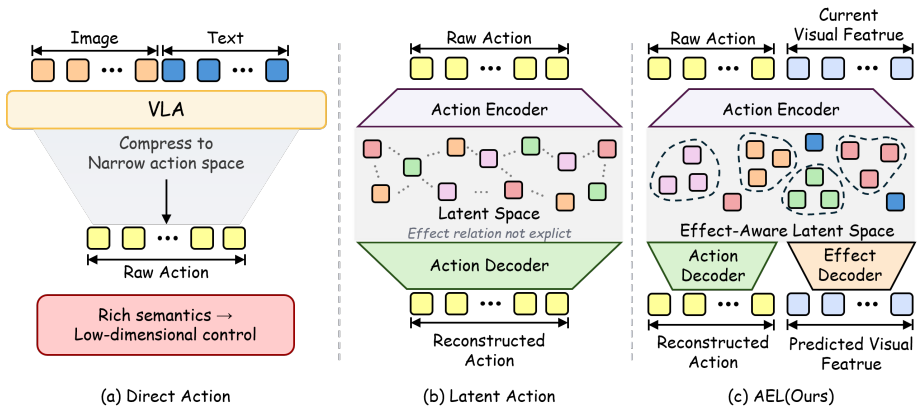

Figure 1: **Motivation of AEL.** (a) **Direct Action:** conventional VLA policies directly map rich multimodal representations to low-level motor commands. (b) **Latent Action:** an intermediate action representation preserves executable control, but action reconstruction alone does not explicitly retain information about action effects. (c) **Action-Effect Latent:** AEL additionally uses the visual state changes induced by executed actions to supervise the latent representation.

To this end, we introduce **AEL (Action-Effect Latents)**, a two-stage framework for VLA control. In Stage I, an action encoder maps the current visual features and an executed action chunk into a scene-conditioned latent, which is trained to reconstruct the action while predicting its semantic visual change over the same horizon. In Stage II, the learned latent becomes the prediction target of a VLA policy conditioned on the current visual observations, language instruction, and robot state, and is decoded into continuous actions. This separation also allows failed or suboptimal interactions to remain useful: while such trajectories may be undesirable imitation targets, their action–effect transitions can still provide supervision for representation learning.

In summary, our contributions are:

(i) We introduce **Action-Effect Latents (AEL)**, which represent executed actions together with their scene-conditioned semantic effects.

(ii) We separate representation learning from policy learning, using future observations only to shape the latent space during Stage I.

(iii) This separation enables failed or suboptimal interactions to provide action–effect supervision without becoming imitation targets.

(iv) Experiments across LIBERO, CALVIN, and real-world single-arm and bimanual manipulation show consistent gains across tasks and embodiments.

2 Related Work

2.1 Vision-Language-Action Models

Vision-Language-Action (VLA) models connect pretrained vision-language representations with robotic control. Early large-scale robot policies such as RT-1 (Brohan et al., 2022) demonstrated scaling across many real-world manipulation tasks, while PaLM-E (Driess et al., 2023) showed how pretrained vision-language representations can be adapted to embodied reasoning. RT-2 (Zitkovich et al., 2023) further casts robot actions as tokens within a vision-language model, enabling knowledge from large-scale visual-language pretraining to transfer to control.

Subsequent work has expanded the scale, diversity, and action parameterization of generalist robot policies. RoboCat (Bousmalis et al., 2024) studies multi-task and multi-embodiment manipulation, while Open X-Embodiment (Open X-Embodiment Collaboration, 2024) enables training across heterogeneous robot datasets. Octo (Ghosh et al., 2024) and OpenVLA (Kim et al., 2024) provide

generalist policies trained on large robot trajectory collections, while RT-H (Belkhale et al., 2024) introduces language-described motions as an intermediate action hierarchy.

Recent VLA systems have also explored more expressive action generators. RDT-1B (Liu et al., 2024) uses diffusion for large-scale bimanual manipulation, whereas π_0 (Black et al., 2024) and $\pi_{0.5}$ (Black et al., 2025) use flow-based action generation. FAST (Pertsch et al., 2025) instead compresses temporally correlated action chunks for autoregressive VLA training.

Despite these differences, most of these policies ultimately learn from demonstrated robot actions as the primary control target. Such supervision specifies the motor trajectory to execute, but does not explicitly use the observed state change produced by that trajectory. AEL complements action supervision with the semantic visual change induced by execution to shape the action representation.

2.2 Latent Action Representations

Latent action models introduce an intermediate representation between observed transitions and low-level robot control, often to exploit videos without robot action labels. LAPA (Ye et al., 2025) infers discrete latent actions from visual transitions and later aligns them with robot actions using a smaller amount of action-labeled data. Genie (Bruce et al., 2024) similarly learns latent actions from unlabeled videos within an action-controllable generative environment.

More recent approaches adapt this idea to generalist robot learning. UniVLA (Bu et al., 2025) learns task-centric latent actions from heterogeneous videos to support transfer across embodiments. JALA (Luo et al., 2026) learns transition-aware latent actions by aligning predictive representations with inverse dynamics and real robot actions. Unified Latent Action (Chen et al., 2026) introduces action-centric cycle consistency across embodiments. ViPRA (Routray et al., 2026) jointly models future observations and motion-centric latent actions before mapping them to continuous control, while Motus (Bi et al., 2026) combines latent actions with video generation, world modeling, and action prediction.

AEL differs in both the source and role of its latent supervision. Rather than inferring the action representation from visual transitions without motor labels, AEL starts from an executed robot action chunk and learns its representation in the context of the current scene. The observed semantic change after execution provides additional supervision, tying the latent to both the executed motor trajectory and its scene-conditioned effect.

2.3 World Models and Predictive Representations

World models learn predictive structure over observations or latent states and use imagined or generated futures for decision making. Dreamer-style methods (Hafner et al., 2025) optimize behavior through imagined latent trajectories. Genie (Bruce et al., 2024) learns action-controllable future dynamics from large-scale video, while UniSim (Yang et al., 2024) learns an interactive real-world simulator for planning and policy training.

Predictive modeling has also been integrated more directly into robot learning. GR-1 (Wu et al., 2024) jointly predicts robot actions and future visual observations. ViPRA (Routray et al., 2026) similarly predicts future observations together with latent actions, while Motus (Bi et al., 2026) unifies video generation, latent-action modeling, and robot action prediction.

AEL uses future observations in a more restricted role. The observed future is used only during Stage I to supervise the representation of an executed action; it is not rolled out for planning or required by the deployed policy. At inference time, Stage II operates solely from the current visual observations, language instruction, and robot state.

3 Method

3.1 Overview

A Vision-Language-Action (VLA) policy maps current visual observations, a language instruction, and robot state to a sequence of robot actions (Zitkovich et al., 2023; Black et al., 2024). AEL introduces an intermediate action representation between multimodal perception and low-level control.

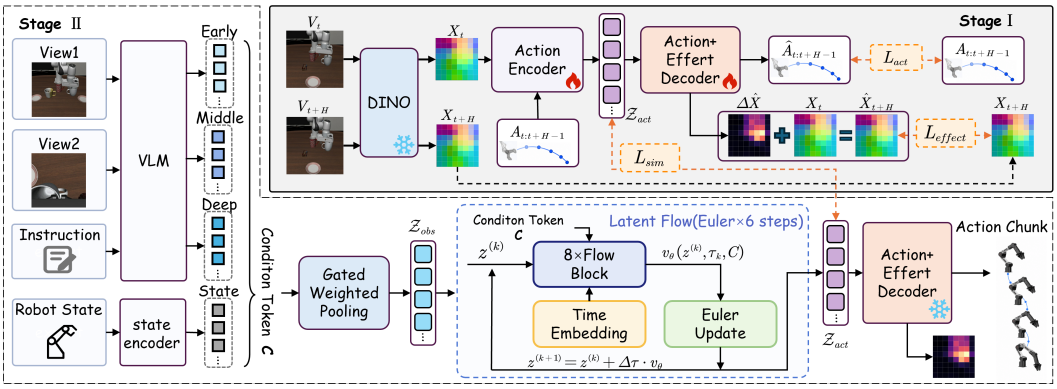

Figure 2: **Overview of AEL.** In Stage I, an executed action chunk and current visual features are jointly encoded into an Action-Effect Latent. Action reconstruction preserves executable control information, while residual effect prediction uses the observed state transition as additional supervision. In Stage II, a VLA policy predicts the learned latent from the current visual observations, language instruction, and robot state, and decodes it into continuous actions.

Rather than defining this representation solely by the executed motor trajectory, we additionally use the visual change observed after execution as scene-dependent supervision.

AEL consists of two stages. Stage I learns an Action-Effect Latent from observed robot interactions. Given the current visual state and an executed action chunk, the latent is trained to preserve the action trajectory while supporting reconstruction of the corresponding future visual representation. Stage II then uses the learned latent as the prediction target of a VLA policy conditioned on the current multimodal context. The predicted latent is decoded back into continuous control. Future observations are therefore used only as Stage I supervision and are not required by the deployed policy.

3.2 Learning Action-Effect Latents

Action-scene representation. Consider a robot trajectory with visual observations V_t and actions a_t. We define an executed action chunk of horizon H and temporal stride s as

$$A_t = [a_t, a_{t+s}, \ldots, a_{t+(H-1)s}],$$

with the corresponding post-execution observation V_{t+Hs}. Each training window is sampled within a single episode to avoid crossing trajectory boundaries.

Rather than reconstructing future RGB observations, we operate in the feature space of a frozen pretrained visual encoder. We extract $X_t = f_{\text{vis}}(V_t)$ and $X_{t+Hs} = f_{\text{vis}}(V_{t+Hs})$. Our main configuration uses the 12th-layer spatial features from a frozen DINOv2-Large encoder (Oquab et al., 2024). Images are resized to 256×256, and the patch features are pooled to an 8×8 spatial grid, yielding $N = 64$ visual tokens. Importantly, X_{t+Hs} is used only as a supervision target and is never provided to the action encoder.

The encoder E_{act} jointly processes the executed action chunk and the current visual state. Action steps and spatial visual features are projected into a common model dimension and augmented with temporal and spatial positional information. A small set of learnable latent tokens is inserted between the two modalities:

$$\begin{aligned} h_i^a &= W_a a_{t+is} + p_i^a + e_a, \qquad h_j^v = W_v x_t^{(j)} + p_j^v + e_v, \\ H^{(0)} &= [\, h_1^a, \ldots, h_H^a,\ r_1, \ldots, r_K,\ h_1^v, \ldots, h_N^v \,], \\ H^{(L)} &= \text{Transformer}(H^{(0)}), \qquad z_{\text{act}} = \phi(H_R^{(L)}). \end{aligned} \tag{1}$$

Here, $H_R^{(L)}$ denotes the final states of the latent tokens and ϕ projects and aggregates them into a fixed-dimensional representation. Joint self-attention allows the latent tokens to integrate information from the executed trajectory and the current spatial context. In our main configuration, we use an action horizon of $H = 8$, temporal stride $s = 1$, and a 1024-dimensional action latent.

Action and effect supervision. We constrain z_{act} through two complementary objectives. First, an action decoder reconstructs the executed action chunk, $\hat{A}_t = D_{\text{act}}(z_{\text{act}})$:

$$\mathcal{L}_{\text{act}} = \frac{1}{H} \sum_{i=0}^{H-1} \|\hat{a}_{t+is} - a_{t+is}\|_2^2 . \tag{2}$$

This objective encourages the latent to retain the low-level information required to recover the executed control trajectory.

Action reconstruction alone does not explicitly use the visual transition produced by execution. We therefore introduce an effect decoder conditioned on both z_{act} and the current visual features. Instead of predicting the future representation independently, the decoder predicts a residual feature update relative to the current representation:

$$\Delta\hat{X}_t = D_{\text{effect}}(z_{\text{act}}, X_t), \qquad \hat{x}_{t+Hs}^{(j)} = \text{normalize}\left(x_t^{(j)} + \Delta\hat{x}_t^{(j)}\right). \tag{3}$$

The reconstructed future representation is supervised against the observed future features using the average token-wise cosine distance:

$$\mathcal{L}_{\text{effect}} = \frac{1}{N} \sum_{j=1}^{N} \left[1 - \cos\left(\hat{x}_{t+Hs}^{(j)}, x_{t+Hs}^{(j)}\right)\right]. \tag{4}$$

The loss is applied to the reconstructed future features rather than directly regressing the residual ΔX. The current representation therefore serves as an explicit reference, while the decoder models the visual-feature change associated with the interaction.

Stage-I objective. The complete representation-learning objective is

$$\mathcal{L}_{\text{Stage-I}} = \mathcal{L}_{\text{act}} + \lambda_{\text{effect}}\mathcal{L}_{\text{effect}}, \tag{5}$$

where $\lambda_{\text{effect}} = 0.2$ in our experiments. The visual encoder remains frozen throughout Stage I. Action reconstruction retains information about the executed trajectory, while future-feature reconstruction introduces additional supervision from the observed state transition.

Stage I is representation learning rather than policy learning: the executed action is available to the encoder when constructing z_{act}. Moreover, Stage I requires only an observed transition (V_t, A_t, V_{t+Hs}), so its interaction data need not coincide exactly with the demonstrations used for imitation in Stage II. This separation allows physically valid but suboptimal interactions to contribute action-effect supervision without necessarily becoming imitation targets, which we examine in Sec. 4.6.

3.3 Latent Policy Learning

After Stage I, the learned Action-Effect Latent becomes the prediction target of the VLA policy. We use Qwen3.5-0.8B as the multimodal backbone throughout our experiments. Given the current visual observations, language instruction, and robot state, the backbone produces multimodal condition features C and an observation representation z_{obs}. The Stage-I encoder provides the target z_{act} corresponding to each demonstrated action chunk.

We model the mapping from the current multimodal context to the target latent using conditional flow matching (Lipman et al., 2023). For $\tau \sim \mathcal{U}(0, 1)$, we define the interpolation

$$z_\tau = (1 - \tau)z_{\text{obs}} + \tau z_{\text{act}}, \tag{6}$$

whose target velocity is $z_{\text{act}} - z_{\text{obs}}$. We train the conditional vector field with

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_\tau \left[\|v_\theta(z_\tau, \tau, C) - (z_{\text{act}} - z_{\text{obs}})\|_2^2\right]. \tag{7}$$

At inference time, the learned flow predicts an action latent $\hat{z}_{\text{act}}$ from the current multimodal context. The action decoder then converts this latent into a continuous action chunk,

$$\hat{A}_t = D_{\text{act}}(\hat{z}_{\text{act}}).$$

The policy therefore operates only on the current multimodal observation at deployment: neither the future observation nor the Stage-I effect-prediction branch is required during inference.

Table 1: Success rates (%) on the four LIBERO suites. "Avg." denotes the average across suites.

Method	Spat.	Obj.	Goal	Long	Avg.
LAPA (Ye et al., 2025)	73.8	74.6	58.8	55.4	65.7
UniVLA (Bu et al., 2025)	96.5	96.8	95.6	92.0	95.2
JALA (Luo et al., 2026)	90.4	96.4	95.2	87.2	92.3
Unified LA (Chen et al., 2026)	97.5	98.2	97.3	93.4	96.6
LA-Direct (Lin et al., 2026)	97.2	98.6	95.8	96.6	97.1
LA-Tok (Lin et al., 2026)	97.0	100.0	92.2	92.6	95.5
Direct Action	95.8	96.2	94.6	90.6	94.3
AEL	**98.4**	**99.2**	**97.8**	95.4	**97.7**

Table 2: Results on CALVIN under the ABC→D protocol.

Method	5 Tasks	Avg. Len.
OpenVLA (Kim et al., 2024)	43.5	3.27
UniVLA (Bu et al., 2025)	56.5	3.80
Unified LA (Chen et al., 2026)	59.4	**3.93**
Direct Action	55.5	3.68
AEL	**60.2**	3.88

4 EXPERIMENTS

We evaluate AEL on two simulation benchmarks and a real-world manipulation platform. We study downstream policy performance, the contribution of effect supervision, the structure of the learned latent representation, and the use of additional suboptimal interactions for representation learning.

4.1 EXPERIMENTAL SETUP

Benchmarks and evaluation. We evaluate on LIBERO, CALVIN, and real-world manipulation tasks. LIBERO consists of four language-conditioned suites: Spatial, Object, Goal, and Long, for which we report success rate on each suite and their average. On CALVIN, we follow the standard ABC→D protocol, training on environments A–C and evaluating on the unseen environment D. We report five-task success and average completed sequence length. For real-world evaluation, each policy is evaluated over 30 trials per task under variations in initial object pose and configuration.

Baselines. We compare with representative latent-action methods, including LAPA (Ye et al., 2025), UniVLA (Bu et al., 2025), JALA (Luo et al., 2026), Unified Latent Action (Chen et al., 2026), and LA-Direct and LA-Tok (Lin et al., 2026). Since these methods differ in backbone, pre-training data, and policy architecture, we use them primarily as published performance references. For controlled comparisons, *Direct Action* uses the same multimodal backbone, demonstrations, action horizon, and optimization budget as AEL while predicting continuous actions directly. *Recon. Latent* uses the same latent-policy pipeline as AEL but learns the action representation using action reconstruction alone, allowing us to isolate the contribution of effect supervision.

Implementation. Stage I uses intermediate spatial features from a frozen DINOv2-Large encoder, with action horizon $H = 8$ and a 1024-dimensional action latent. We set $\lambda_{\text{effect}} = 0.2$ and train the effect branch using cosine supervision on the reconstructed future feature. Experiments are implemented in PyTorch on a server equipped with two NVIDIA RTX 4090 GPUs, with each Stage I run using a single GPU. Unless otherwise stated, controlled variants use the same training data and optimization budget. Additional architecture and training details are provided in the appendix.

4.2 BENCHMARK RESULTS

Tables 1 and 2 summarize the main simulation results. For LIBERO, we report success rates on all four task suites together with their average, following the reporting format commonly used by recent latent-action methods. For CALVIN, we report five-task success and average completed sequence length.

On LIBERO, AEL achieves an average success rate of 97.7%, improving the matched Direct Action baseline by 3.4 percentage points. The improvement is consistent across all four suites, with the largest gain on LIBERO-Long, where success increases from 90.6% to 95.4%. AEL also performs competitively with recent latent-action methods, although these published results use different training data, backbones, and policy architectures.

AEL shows a similar improvement over the matched baseline on CALVIN. Five-task success increases from 55.5% to 60.2%, while the average completed sequence length improves from 3.68 to

Table 3: Controlled ablation of AEL. "Future" denotes supervision from the future visual representation, while "Residual" denotes predicting the future representation as an action-conditioned change relative to the current visual feature. LIBERO reports average success rate (%), and CALVIN reports average completed sequence length.

Variant	Latent	Future	Residual	LIBERO	CALVIN
Direct Action	–	–	–	94.3	3.68
Recon. Latent	Yes	–	–	95.1	3.72
+ Future Feature	Yes	Yes	–	96.2	3.76
AEL	Yes	Yes	Yes	**97.7**	**3.88**

3.88. The consistent gain over Direct Action on both benchmarks indicates that the benefit of AEL is not limited to the LIBERO setting.

We next conduct controlled ablations to separate the contributions of the latent action representation and visual-effect supervision.

4.3 Controlled Ablations

We conduct controlled ablations to isolate the contributions of the latent action representation and the visual-effect supervision introduced in Stage I. All variants use the same multimodal backbone, demonstrations, action horizon, and optimization budget. For latent-action variants, the same Stage II latent policy is used, so the comparison focuses on how the Stage I representation is constructed.

Introducing a reconstruction-based action latent provides a modest but consistent improvement over direct action prediction. On LIBERO, average success increases from 94.3% to 95.1%, while CALVIN average sequence length improves from 3.68 to 3.72. This suggests that introducing an intermediate action representation is already beneficial under the matched policy setting.

We next add supervision from the observed future visual representation. The + *Future Feature* variant improves LIBERO performance to 96.2% and CALVIN average sequence length to 3.76, indicating that the state observed after execution provides useful information beyond action reconstruction alone.

AEL further changes how this visual supervision is formulated. Instead of predicting the future representation independently, AEL models it as a residual effect relative to the current visual state, $\hat{X}_{t+H} \approx X_t + \Delta\hat{X}_t$. This yields 97.7% average success on LIBERO and 3.88 average sequence length on CALVIN, providing the largest improvement among the controlled variants. The comparison suggests that representing the observed transition as an action-conditioned change from the current scene provides a more useful training signal than direct future-feature prediction.

Overall, the ablation shows a consistent progression from direct action prediction to reconstruction-based latent learning, future visual supervision, and finally residual effect modeling. These results support the two central components of Stage I: preserving executable action information while using the observed scene transition to shape the learned action representation.

Downstream policy performance alone, however, does not reveal what information is encoded in the latent space. We therefore next analyze the learned representation directly.

4.4 Action-Effect Representation Analysis

We further examine the Stage I representation on held-out trajectories. Table 4 reports action reconstruction and future-feature prediction metrics for the controlled Stage I variants. The reconstruction-only latent achieves the lowest action MSE, while AEL remains close at 0.05898. Compared with direct future-feature prediction, AEL reduces the future-feature cosine error from 0.004895 to 0.004471. Moreover, its prediction improves over simply copying the current feature on 99.12% of the validation windows, compared with 71.43% for the direct future-feature variant. These results suggest that the residual effect formulation improves future-feature modeling while retaining substantial information about the executed action.

Table 4: Stage-I representation analysis. Gain > 0 denotes the fraction of samples for which the predicted future feature improves over copying the current feature.

Metric	Recon.	Future	AEL
Action MSE ↓	**0.05693**	0.06289	0.05898
Future cosine ↓	–	0.004895	**0.004471**
Gain > 0 ↑	–	71.43%	**99.12%**

Table 5: Effect of successful (S) and suboptimal (U) interactions across the two training stages on real-world bimanual tasks. Suboptimal interactions use a 20% erroneous-action insertion rate.

Stage I	Stage II	Bimanual Avg.
S	S	57.8
S+U	S	**61.1**
S	S+U	42.2
S+U	S+U	43.3

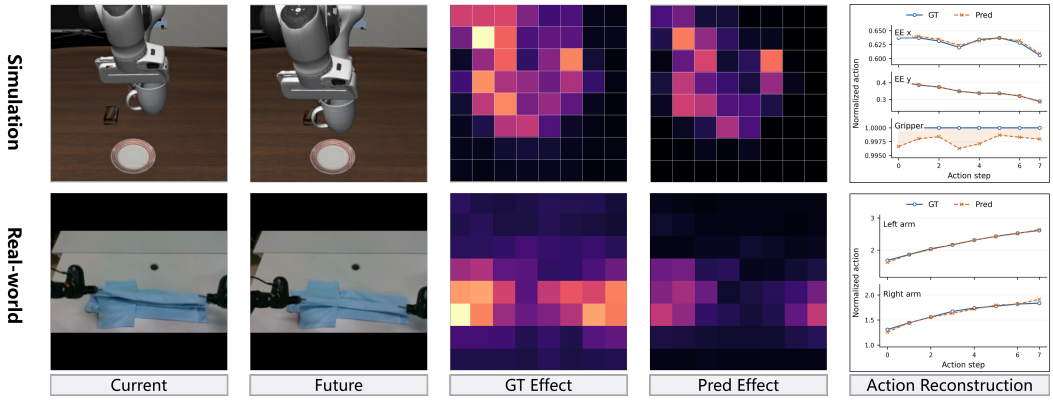

Figure 3: **Qualitative action-effect decoding.** The top row shows a simulation example and the bottom row a real-world example. From left to right, we show the current observation, the post-execution observation, the observed visual-feature change, the visual-feature change predicted from z_{act}, and representative action dimensions reconstructed from the same latent. Only representative action dimensions are shown for clarity.

Qualitative action-effect decoding. Figure 3 complements the quantitative analysis with examples of the information decoded from z_{act}. The predicted effect maps qualitatively follow the spatial changes observed after the action chunk is executed, while the reconstructed action dimensions remain close to the executed trajectory. Action reconstruction is used here only as a Stage I representation diagnostic: since the encoder receives the executed action as input, the reconstructed trajectories should not be interpreted as policy predictions.

4.5 Real-World Evaluation

We evaluate AEL on six real-world manipulation tasks using a dataset collected on our own robot platform. The dataset includes three single-arm tasks (*Pick-and-Place*, *Mug Placement*, and *Lid Closing*) and three bimanual tasks (*Object Handover*, *Container Packing*, and *Garment Folding*). All compared methods are trained on the same collected demonstrations and evaluated under the same protocol, with 30 trials per task. We compare AEL with Diffusion Policy (DP) and π_0.

As shown in Figure 4, AEL achieves average success rates of 74.4% on the single-arm tasks and 61.1% on the bimanual tasks. Under the same evaluation protocol, π_0 obtains 63.3% and 50.0%, while DP obtains 50.0% and 37.8%, respectively. Across all six tasks, the corresponding average success rates are 67.8% for AEL, 56.7% for π_0, and 43.9% for DP. These results indicate that the performance trend observed in simulation also extends to the real-world setting with independently collected robot data.

Figure 5 provides a qualitative visualization of the visual-feature changes predicted by the Stage-I effect decoder on real-world trajectories. The deployed Stage-II policy does not use the effect-prediction branch.

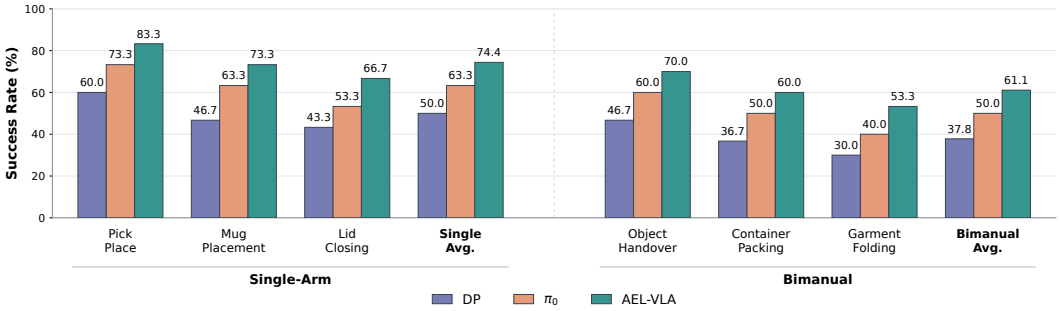

Figure 4: **Real-world manipulation performance.** Success rates over 30 trials per task on three single-arm and three bimanual manipulation tasks.

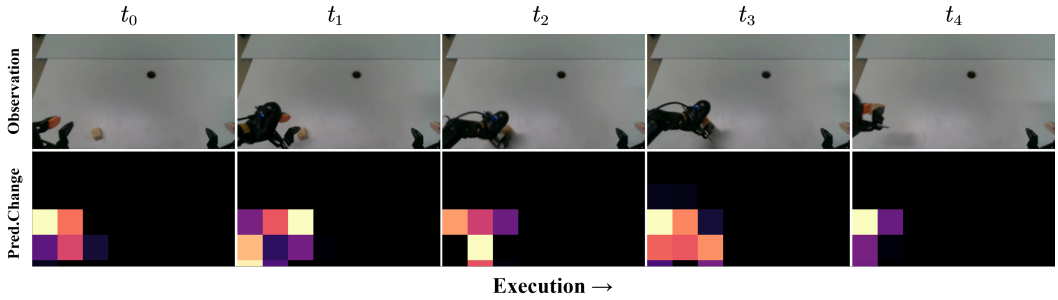

Figure 5: **Real-world action-effect visualization.** Observations from a real-world execution (top) and the corresponding visual-feature changes predicted by the Stage-I effect decoder (bottom). The visualization is used only for representation analysis and is not part of policy inference.

4.6 Suboptimal Interaction Study

AEL separates representation learning from imitation-based policy learning. This allows physically valid but suboptimal interactions to provide action-effect supervision without necessarily treating them as desirable imitation targets. We construct suboptimal interactions using a 20% erroneous-action insertion rate and evaluate different ways of using them across the two stages on the three real-world bimanual tasks.

As shown in Table 5, using successful demonstrations in both stages gives an average success rate of 57.8%. Adding suboptimal interactions only to Stage I increases the average to 61.1%. In contrast, including suboptimal interactions in Stage II reduces the success rate to 42.2%, while using them in both stages gives 43.3%. These results suggest that, in this setting, suboptimal interactions can provide useful action-effect supervision for representation learning, whereas directly including the same interactions as imitation targets can degrade policy performance.

5 Conclusion

We presented AEL, a two-stage framework for learning Action-Effect Latents for vision-language-action policies. Rather than representing robot actions solely through their control trajectories, AEL incorporates the observed visual consequences of executed actions as an additional supervision signal for learning the action representation. In Stage I, the latent is trained to preserve continuous action information while reconstructing the future visual representation through a residual effect formulation. In Stage II, a VLA policy learns to infer this latent from the current multimodal context through conditional latent flow and decode it into continuous control. Future observations are used only as supervision during representation learning and are not required by the deployed policy.

The current study is conducted under a relatively modest compute budget, with our experiments primarily focusing on the method itself and its behavior at the present scale. Future work could investigate AEL with stronger pretrained backbones, larger interaction datasets, longer-horizon behaviors, and more diverse robot embodiments, enabling a more systematic study of the scaling and generalization properties of action-effect representations.

REFERENCES

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

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Manon Devin, Alex X. Lee, Maria Bauza Villalonga, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Fernandes Martins, et al. Robocat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*, 2024.

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

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 2024.

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

Guangyan Chen, Qi Shao, Te Cui, Zichen Zhou, Weixin Mao, Luojie Yang, Meiling Wang, Yi Yang, Hua Chen, and Yufeng Yue. Learning a unified latent action space from videos with action-centric cycle consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12871–12880, 2026.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 2023.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.120.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Yihan Lin, Haoyang Li, Yang Li, Haitao Shen, Yihan Zhao, Chao Shao, and Jing Zhang. From pixels to tokens: A systematic study of latent action supervision for vision-language-action models. *arXiv preprint arXiv:2605.04678*, 2026.

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

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: A diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

Hao Luo, Ye Wang, Wanpeng Zhang, Haoqi Yuan, Yicheng Feng, Haiweng Xu, Sipeng Zheng, and Zongqing Lu. Joint-aligned latent action: Towards scalable vla pretraining in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35047–35058, 2026.

Open X-Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models. In *IEEE International Conference on Robotics and Automation*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

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

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736. PMLR, 2023.

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

Sherry Yang, Yilun Du, Seyed Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024.

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

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

APPENDIX

A AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, as assistive tools during the preparation of this work. Their use included manuscript editing and translation, improving the organization and clarity of the presentation, discussing and refining experimental and analysis protocols, assisting with the interpretation and presentation of experimental results, and supporting the implementation, debugging, and checking of parts of the experimental and analysis code.

Generative AI tools were not used to collect experimental data, generate experimental results, or create the robot observations and experimental images reported in this paper. All simulation and real-world observations shown in the paper were obtained from our actual experiments. All reported numerical results were computed from experimental runs and recorded data.

All AI-assisted code was reviewed and tested by the authors before being used in experiments. Experimental configurations, methodological choices, evaluation protocols, and scientific conclusions were determined and verified by the authors. All AI-assisted text and code were reviewed and revised as necessary, and the authors take full responsibility for the accuracy and content of the final manuscript.

B IMPLEMENTATION DETAILS

B.1 STAGE-I ACTION-EFFECT REPRESENTATION LEARNING

Visual representation. We use a frozen DINOv2-Large encoder to obtain spatial visual features from its 12th Transformer layer. Input images are resized to 256×256. The resulting patch features are spatially pooled to an 8×8 grid, yielding $N = 64$ visual tokens with dimension 1024. The DINOv2 encoder remains frozen throughout Stage I training.

Action-scene encoder. We use an action horizon of $H = 8$ with temporal stride $s = 1$. Each action in the chunk is projected into the encoder hidden space and augmented with temporal positional information and an action-type embedding. Visual tokens are projected in the same manner and augmented with spatial positional information and a visual-type embedding. The action and visual tokens are jointly processed together with learnable latent tokens using bidirectional self-attention.

The action-scene encoder contains four Transformer layers with hidden dimension 512 and eight attention heads. The output latent tokens are aggregated and projected to produce the 1024-dimensional Action-Effect Latent z_{act}. Importantly, future visual observations are never provided as inputs to this encoder; they are used only to construct the Stage I supervision signal.

Action reconstruction. An action decoder reconstructs the executed H-step action chunk from z_{act}. We supervise this branch using mean squared error between the reconstructed and executed actions. This objective encourages the learned latent to retain information necessary for recovering the underlying motor trajectory.

Visual-effect supervision. In parallel, an effect decoder receives the learned latent together with the current visual representation and predicts a residual feature update:

$$\Delta \hat{X}_t = D_{\text{effect}}(z_{\text{act}}, X_t). \tag{8}$$

The predicted future feature at spatial position j is reconstructed as

$$\hat{x}^{(j)}_{t+Hs} = \text{normalize}\left(x^{(j)}_t + \Delta \hat{x}^{(j)}_t\right). \tag{9}$$

The reconstruction is supervised using the average token-wise cosine error,

$$\mathcal{L}_{\text{effect}} = \frac{1}{N}\sum_{j=1}^{N}\left[1 - \cos\left(\hat{x}^{(j)}_{t+Hs}, x^{(j)}_{t+Hs}\right)\right], \tag{10}$$

where $N = 64$. We use uniform weighting across spatial tokens and do not directly regress the feature difference ΔX itself.

The complete Stage I objective is

$$\mathcal{L}_{\text{Stage-I}} = \mathcal{L}_{\text{act}} + 0.2\,\mathcal{L}_{\text{effect}}. \tag{11}$$

Window construction. Training windows are always constructed within a single episode. An action chunk starting at time t is paired with the corresponding current visual representation X_t and the visual representation after the execution horizon, X_{t+Hs}. Windows that would cross an episode boundary are discarded. The same episode-level constraint is used when constructing validation samples.

B.2 Stage-II Latent Policy Learning

Multimodal backbone. We use Qwen3.5-0.8B as the multimodal backbone throughout our experiments. Rather than directly predicting embodiment-specific continuous actions, Stage II predicts the 1024-dimensional Action-Effect Latent learned in Stage I. The predicted latent is then decoded into an $H = 8$ continuous action chunk using the Stage I action decoder.

Latent prediction. For each demonstration window, the frozen Stage I encoder provides the target latent z_{act}. A conditional latent flow is trained to predict this target from the current multimodal context. The conditioning information contains the current visual observations, language instruction, and robot state. This preserves a separation between representation learning in Stage I and policy learning in Stage II: executed actions are available to the Stage I encoder during representation learning, but are not available to the policy when predicting the latent at deployment.

During Stage II training, the Stage I action-effect encoder and action decoder are kept fixed. At inference time, the policy first predicts $\hat{z}_{\text{act}}$ from the current context and then maps it to a continuous action chunk,

$$\hat{A}_t = D_{\text{act}}\left(\hat{z}_{\text{act}}\right). \tag{12}$$

Neither the future observation nor the Stage I effect decoder is required during policy execution.

Real-world inputs. The robot platform records one top-view stream and two wrist-view streams, together with robot state and continuous actions. Stage II conditions latent prediction on the visual observations used by the corresponding policy configuration, together with the language instruction and current robot state. All visual inputs correspond to the current control step; no post-execution observations are provided to the deployed policy.

B.3 Optimization and Compute

For Stage I, the training budget scales with the number of tasks. For a training set containing T tasks, we train for $10{,}000 \times T$ optimization steps, corresponding to an additional 10,000 optimization steps for each additional task. We use AdamW with a batch size of 16, an initial learning rate of 10^{-4}, a minimum learning rate of 10^{-5}, 1,000 warm-up steps, and weight decay of 0.01.

For the real-world Stage II experiments, the latent policy is trained for 30,000 optimization steps. The effective batch size is 8, implemented with a micro-batch size of 2 and gradient accumulation over four optimization micro-steps.

Our implementation uses PyTorch 2.11. Experiments are conducted on a server equipped with two NVIDIA RTX 4090 GPUs. Each Stage I training run uses one GPU. Controlled Stage I variants use the same data construction procedure and training budget as the corresponding AEL model.

C Real-World Experimental Setup

C.1 Robot Platform and Observations

Our real-world experiments use a dual-arm manipulation platform composed of two Piper robot arms. The complete bimanual action representation is 14-dimensional and jointly describes the continuous control commands for the two arms.

Visual observations are captured using three Intel RealSense D435i cameras. One camera is mounted above the workspace to provide a global top view, while two additional D435i cameras are mounted near the wrists of the two robot arms to provide egocentric views of local manipulation. The three camera streams, robot states, and control signals are recorded synchronously at 30 Hz.

The real-world benchmark contains three single-arm tasks and three bimanual tasks. The single-arm tasks are *Pick-and-Place*, *Mug Placement*, and *Lid Closing*. The bimanual tasks are *Object Handover*, *Container Packing*, and *Garment Folding*. This collection covers both localized single-arm manipulation and tasks requiring coordination between the two arms.

C.2 Data Collection

The real-world dataset is collected on our own robot platform. Each recorded episode contains synchronized top-view and dual wrist-view observations, robot state, and the corresponding continuous action sequence. The dataset is stored as complete trajectories rather than as independently shuffled frames.

Training and validation data are separated at the episode level. Consequently, frames or action windows originating from the same trajectory are not split between training and validation sets. For Stage I, an individual training sample consists of a current visual observation, an $H = 8$ action chunk, and the observation following the same execution horizon. Samples are discarded whenever the complete temporal window cannot be contained within a single episode.

This construction is particularly important for action-effect supervision, since the future observation must correspond to the actual consequence of the associated action chunk rather than to an observation from another trajectory.

Suboptimal interactions. For the suboptimal-interaction study, we additionally construct interaction data containing erroneous actions. The erroneous-action insertion rate is 30%. These trajectories are used to separately study whether suboptimal interactions are beneficial when used as Stage I action-effect supervision and whether the same interactions remain beneficial when directly included as Stage II imitation targets. Successful and suboptimal interactions are denoted by S and U, respectively.

C.3 Evaluation Protocol

Each of the six real-world tasks is evaluated over 30 trials. We use binary task success as the evaluation metric: a trial contributes one success only when the task-specific completion condition is satisfied during the evaluation rollout. Success rates are first computed independently for each task and are then averaged across the three single-arm tasks and the three bimanual tasks. The overall real-world score averages performance across all six tasks. This results in 180 evaluation trials per policy across the full real-world benchmark.

For downstream evaluation, the compared policies are adapted using the same task demonstrations where their respective architectures permit. Because the compared methods differ in architecture and pretraining, the real-world baseline comparison is treated as a performance comparison rather than an architecture-matched controlled ablation.

For the suboptimal-interaction experiment, we evaluate only the three bimanual tasks. Each task again contains 30 trials, giving 90 trials for each training configuration. We evaluate four combinations: S/S, $S{+}U/S$, $S/S{+}U$, and $S{+}U/S{+}U$, where the terms before and after the slash indicate the interaction data used for Stage I and Stage II, respectively.

D DINO Layer Selection

We additionally study the choice of DINOv2 feature layer used for Stage I. We compare features extracted from layers 4, 8, 12, and 24 of the frozen DINOv2-Large encoder. All variants in this controlled study are trained for 10,000 optimization steps and evaluated on the same complete validation set.

Table 6: Comparison of different DINOv2-Large feature layers under the same 10,000-step training budget and validation set. Layer 24 corresponds to the final Transformer layer. Because different DINO layers define different feature spaces, cosine-based quantities should not be compared directly across rows. This study was conducted with an earlier change-weighted Stage-I configuration and is used only to motivate the choice of feature layer.

DINO Layer	Action MSE ↓	Future Cosine ↑	Copy Cosine	Cosine Gain ↑	Shuffle Gap ↑
4	0.14100	0.996368	0.994431	0.001937	0.0000683
8	0.13538	0.995220	0.992797	0.002424	0.0000422
12	**0.06240**	0.995484	0.993155	0.002329	0.0000410
24	0.09797	0.975931	0.965580	0.010351	0.0000863

Among the evaluated layers, the 12th-layer features yield the lowest action reconstruction error, with an Action MSE of 0.06240. This is approximately 36.3% lower than the 0.09797 MSE obtained using the final, 24th layer. Based on this result, we use the 12th-layer DINOv2 features throughout the main experiments.

The cosine-based quantities in Table 6 characterize prediction quality within the feature space of each individual layer. However, their absolute magnitudes are not directly comparable across layers, since each layer defines a different representation space with different feature statistics. We therefore do not use the larger cosine gain of the 24th layer as evidence that it provides a better Stage-I representation.

This layer-selection experiment predates the final AEL configuration and was conducted using an earlier change-weighted visual-effect objective. The additional spatial-correlation, Top-k overlap, and latent-swap diagnostics reported for the final ablations were evaluated only on the 12th-layer checkpoints and are therefore not used for cross-layer comparison.

E Training and Inference Pseudocode

For completeness, Algorithms 1 and 2 summarize the two-stage training procedure of AEL. Future observations are used only in Stage I to construct visual-effect supervision and are not required by the Stage II policy at inference time.

Algorithm 1 Stage-I Action-Effect Representation Learning

Require: Episode dataset $\mathcal{D}$; frozen DINOv2-Large encoder E_{vis}; action-scene encoder E_{AE}; action decoder D_{act}; effect decoder D_{effect}; horizon $H = 8$; stride $s = 1$

for each optimization step **do**
  Sample an episode $\tau \sim \mathcal{D}$
  Sample time t such that $(t, \ldots, t + Hs)$ remains inside τ
  Extract executed action chunk $A_t = \{a_t, \ldots, a_{t+H-1}\}$
  Extract current and future observations o_t and o_{t+Hs}
  $X_t \leftarrow E_{\text{vis}}(o_t)$
  $X_{t+Hs} \leftarrow E_{\text{vis}}(o_{t+Hs})$
  Pool spatial features to 8×8 tokens
  $z_{\text{act}} \leftarrow E_{\text{AE}}(A_t, X_t)$
  $\hat{A}_t \leftarrow D_{\text{act}}(z_{\text{act}})$
  $\mathcal{L}_{\text{act}} \leftarrow \text{MSE}(\hat{A}_t, A_t)$
  $\Delta\hat{X}_t \leftarrow D_{\text{effect}}(z_{\text{act}}, X_t)$
  for $j = 1, \ldots, 64$ **do**
    $\hat{x}^{(j)}_{t+Hs} \leftarrow \text{normalize}\left(x^{(j)}_t + \Delta\hat{x}^{(j)}_t\right)$
  end for
  $\mathcal{L}_{\text{effect}} \leftarrow \frac{1}{64}\sum_{j=1}^{64}\left[1 - \cos\left(\hat{x}^{(j)}_{t+Hs}, x^{(j)}_{t+Hs}\right)\right]$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{act}} + 0.2\,\mathcal{L}_{\text{effect}}$
  Update E_{AE}, D_{act}, and D_{effect} using $\nabla\mathcal{L}$
end for

Algorithm 2 Stage-II Latent Policy Learning and Inference

Require: Demonstration dataset $\mathcal{D}$; frozen Stage-I encoder E_{AE}; frozen action decoder D_{act}; multimodal backbone M; conditional latent flow F_θ

Training:
for each optimization step **do**
  Sample a demonstration window from an episode
  Obtain current multimodal context $c_t = (o_t, l, q_t)$
  Obtain demonstrated action chunk A_t
  Extract current visual representation X_t
  $z_{\text{act}} \leftarrow E_{\text{AE}}(A_t, X_t)$
  $h_t \leftarrow M(c_t)$
  Compute conditional flow-matching loss $\mathcal{L}_{\text{flow}} \leftarrow \text{CFM}(F_\theta, h_t, z_{\text{act}})$
  Update the trainable Stage-II parameters using $\nabla\mathcal{L}_{\text{flow}}$
end for

Inference:
Observe current multimodal context $c_t = (o_t, l, q_t)$
$h_t \leftarrow M(c_t)$
$\hat{z}_{\text{act}} \leftarrow \text{FlowSample}(F_\theta, h_t)$
$\hat{A}_t \leftarrow D_{\text{act}}(\hat{z}_{\text{act}})$
return predicted continuous action chunk $\hat{A}_t$