# BEYOND ACTIONS: LEARNING FUTURE OPERATION TARGETS FOR VISION-LANGUAGE-ACTION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Actions specify what a robot should execute, but do not describe what those commands will achieve. Vision-language-action (VLA) policies nevertheless learn primarily by predicting action chunks from images, language, and state. We introduce Action–Future Coupled Effect Learning (AFCE), which makes a future-grounded physical Effect the policy's primary prediction target. An offline tokenizer constructs temporally ordered Effect tokens from demonstrated actions, realized robot responses, and sparse visual transitions. Reconstruction objectives encourage the representation to retain executable detail and observed physical change. A flow-matching VLA predicts Effects from current images, instruction, and robot state, while a state-conditioned decoder converts them into action chunks. Embodiment-specific interfaces allow single- and dual-arm tasks to share the Effect model. On DexJoCo, AFCE achieves $50.97\%$ overall success, improving over $\pi_{0.5}$ and UVT by 11.82 and 15.21 percentage points, respectively. It leads on all five bimanual tasks, highlighting the value of Effect supervision for coordinated dexterous manipulation.

## 1 INTRODUCTION

Vision-language-action (VLA) models have made substantial progress toward generalist robot control by mapping images, language, and robot state to continuous actions (Zitkovich et al., 2023; Octo Model Team et al., 2024; Kim et al., 2025; Black et al., 2024; 2025). Yet the target used to train this mapping usually describes *what the robot commanded*, not *what the command changed*. An action sequence specifies motor behavior, but leaves its effects on the robot and environment implicit. In dexterous manipulation, this gap matters: contact, hand configuration, and object pose can turn similar commands into different outcomes, while different motions can achieve the same task-relevant change. The challenge is to make physical progress part of the policy's prediction target without sacrificing the detail needed to execute an action.

Predicting the future offers one way to expose such progress. World and world-action models use future observations or latent dynamics to support control (Wu et al., 2024; Cen et al., 2025; Zhang et al., 2025; Ye et al., 2026). However, a complete visual future contains changes due to appearance, occlusion, and camera motion that need not determine the next control command. Conversely, a scene change alone may leave out the contact timing, finger coordination, and robot motion required to produce it. Latent-action and future-informed visuomotor methods provide more compact targets (Ye et al., 2024; Bu et al., 2025; Chen et al., 2026; Luo et al., 2026a; Liu et al., 2026; Xu et al., 2026; Luo et al., 2026b), but compactness alone does not resolve whether a target is both predictable from the current observation and sufficient for precise action decoding.

We therefore seek a target that connects three requirements: it must be predictable before acting, retain task-relevant information about the resulting physical transition, and support recovery of executable continuous actions. We introduce **AFCE (Action–Future Coupled Effect Learning)**, which learns a temporal *Effect* from synchronized demonstrations of executed actions, realized robot-state changes, and sparse world changes. Rather than specifying the Effect through hand-designed semantic labels, we let these sources interact and constrain a shared representation through joint reconstruction. The action objective retains control detail, while the robot-response and world-change objectives ground that detail in observed consequences.



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.

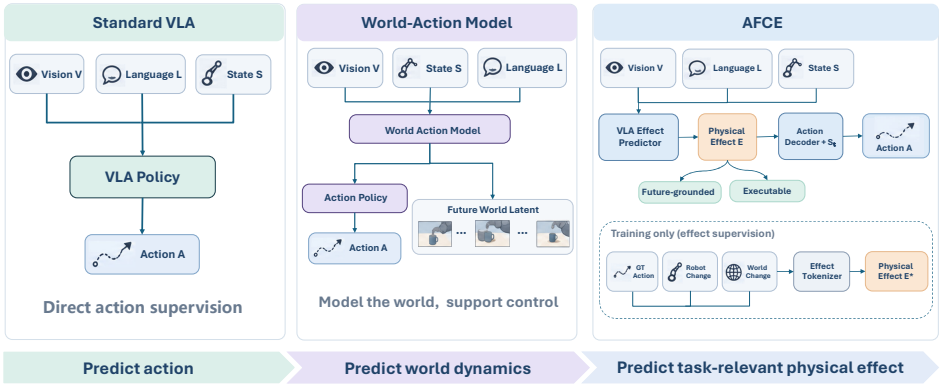



Figure 1: Comparison of policy supervision paradigms. A standard VLA predicts actions directly; world-action models use future dynamics to support control; AFCE predicts an action-decodable Effect grounded in executed actions, robot response, and world change. Future evidence constructs training targets but is unavailable to the deployed policy.

Figure 1 illustrates the resulting training interface. An offline tokenizer constructs Effect targets from recorded trajectories, where future evidence is available. The VLA then learns to predict these targets using only the current images, language instruction, and robot state. At deployment, a decoder maps the predicted Effect and current state to an action chunk. Thus, future observations shape the supervision without becoming inputs to the deployed policy.

The central test of this interface is whether the Effect remains useful *after prediction*. Accurate reconstruction from a ground-truth Effect does not ensure that the VLA can predict that Effect, nor that decoded actions succeed under closed-loop control. We evaluate these questions separately: reconstruction tests what the representation retains; ground-truth-Effect replay tests whether it supports execution; and predicted-Effect rollout tests the complete policy against direct action supervision under the same evaluation protocol. In mixed single- and dual-arm training, AFCE improves aggregate task success, particularly on dual-arm tasks, while some fine-control tasks remain challenging.

Our contributions are:

- We formulate VLA target learning around three coupled requirements: predictability from current observations, information about task-relevant future changes, and continuous-action executability.
- We introduce AFCE, a temporal Effect learned by jointly reconstructing executed actions, realized robot response, and sparse world change from a shared representation.
- We evaluate the representation and the complete policy through reconstruction, ground-truth-Effect replay, and predicted-Effect closed-loop control, showing aggregate gains under mixed single- and dual-arm training while identifying remaining fine-control limitations.

## 2 Related Work

### 2.1 VLA Policies and Action Supervision

Vision-language-action (VLA) models have established a powerful interface between semantic understanding and robot control. RT-2 casts discretized actions as language-like tokens, while Octo and OpenVLA learn generalist policies from heterogeneous robot data (Zitkovich et al., 2023; Octo Model Team et al., 2024; Kim et al., 2025). $\pi_0$ and $\pi_{0.5}$ further combine pretrained visual-language representations with a flow-matching action expert for continuous control (Black et al., 2024; 2025). Complementary advances such as ACT, Diffusion Policy, and FAST improve action prediction through chunking, expressive generative modeling, and efficient tokenization (Zhao et al., 2023; Chi et al., 2023; Pertsch et al., 2025). Together, these methods provide increasingly capable ways to generate robot commands. They also motivate a closer examination of what the policy is asked to predict: an action records the commanded motion, whereas the resulting robot and scene changes

carry additional information about the intended manipulation. Building on these advances, AFCE preserves the VLA observation interface and continuous action execution while introducing a physical Effect as the policy's intermediate supervision target.

### 2.2 World Models and Future-Informed Control

Future-aware learning offers a complementary route to stronger visuomotor policies. GR-1 and VidMan use video dynamics to inform manipulation, while WorldVLA and DreamZero connect predictions about the evolving world to action generation (Wu et al., 2024; Wen et al., 2024; Cen et al., 2025; Ye et al., 2026). FutureVLA and Being-H0.7 further show how future-informed representations can guide policies that act from current observations (Xu et al., 2026; Luo et al., 2026b). Collectively, these studies establish future information as a valuable training signal for control. AFCE asks how that signal can be organized into a target that is both informative about physical change and directly useful for producing actions. Demonstrated actions, realized robot responses, and sparse visual transitions jointly specify the Effect during training; at deployment, the policy predicts it from the current context and a state-conditioned decoder converts it into robot commands. This formulation explicitly links future-grounded supervision to the action interface used for closed-loop execution.

### 2.3 Latent Actions and Transferable Policy Targets

Latent-action learning provides intermediate targets that capture structure beyond individual action coordinates. LAPA derives latent actions from visual transitions, while UniVLA and villa-X develop representations for learning from robot and video data (Ye et al., 2024; Bu et al., 2025; Chen et al., 2026). JALA and LARA connect transition-derived features to actions and policy learning (Luo et al., 2026a; Liu et al., 2026). UVT combines demonstrated action chunks with discrete visual dynamics codes into a decodable policy target (Feng & Jain, 2026). AFCE develops this direction by constructing a temporal Effect from demonstrated commands, realized robot responses, and sparse visual transitions. Its action and response branches relate observed scene changes to both the issued commands and the motion actually realized. Realized responses are particularly informative in dexterous manipulation, where contact and hand configuration can make similar commands produce different outcomes. Temporal Effect tokens preserve the progression from initial contact to subsequent object interaction. A state-conditioned decoder maps the Effect to native actions, making the learned target useful for continuous control. Embodiment-specific adapters accommodate different native state and action dimensions while retaining a common Effect shape.

## 3 Action–Future Coupled Effect Learning

We consider language-conditioned manipulation with current observation $\mathbf{o}_t = (\mathbf{I}_t, \ell, \mathbf{s}_t)$, where $\mathbf{I}_t$ denotes the images, $\ell$ the instruction, and $\mathbf{s}_t$ the robot state. A conventional VLA predicts an action chunk $\mathbf{A}_t = [\mathbf{a}_t, \dots, \mathbf{a}_{t+H-1}]$ directly from $\mathbf{o}_t$. AFCE instead trains the policy to predict a temporal *Effect*: a compact target constructed from issued actions, the robot's realized response, and changes in the observed world. The complete framework is shown in Figure 2.

For each demonstration window, let $\Delta\mathbf{S}_t$ denote the realized robot-response sequence and $\mathbf{V}_t$ sparse visual transition evidence over $[t, t+H]$. Stage I constructs

$$\mathbf{E}_t^* = T_\phi(\mathbf{A}_t, \Delta\mathbf{S}_t, \mathbf{V}_t) \in \mathbb{R}^{H \times D_E}, \tag{1}$$

where $T_\phi$ is the Effect tokenizer. We use $H = 30$ and $D_E = 256$, yielding one ordered token per action step. The superscript $*$ distinguishes a target obtained from demonstration evidence from an Effect predicted at deployment.

Stage I learns the tokenizer with action and transition reconstruction. We then freeze it, cache its Effect outputs, and use fixed statistics to obtain normalized targets $\mathbf{Z}_t^*$ for Stage II. The VLA predicts these targets from $\mathbf{o}_t$ while the action decoder adapts to predicted Effects. At deployment, the policy samples a normalized Effect and decodes its denormalized counterpart into actions:

$$\widehat{\mathbf{Z}}_t \sim p_\theta(\mathbf{Z} \mid \mathbf{o}_t), \qquad \widehat{\mathbf{A}}_t = D_\psi\Big(\mathcal{N}^{-1}(\widehat{\mathbf{Z}}_t), \mathbf{s}_t\Big). \tag{2}$$

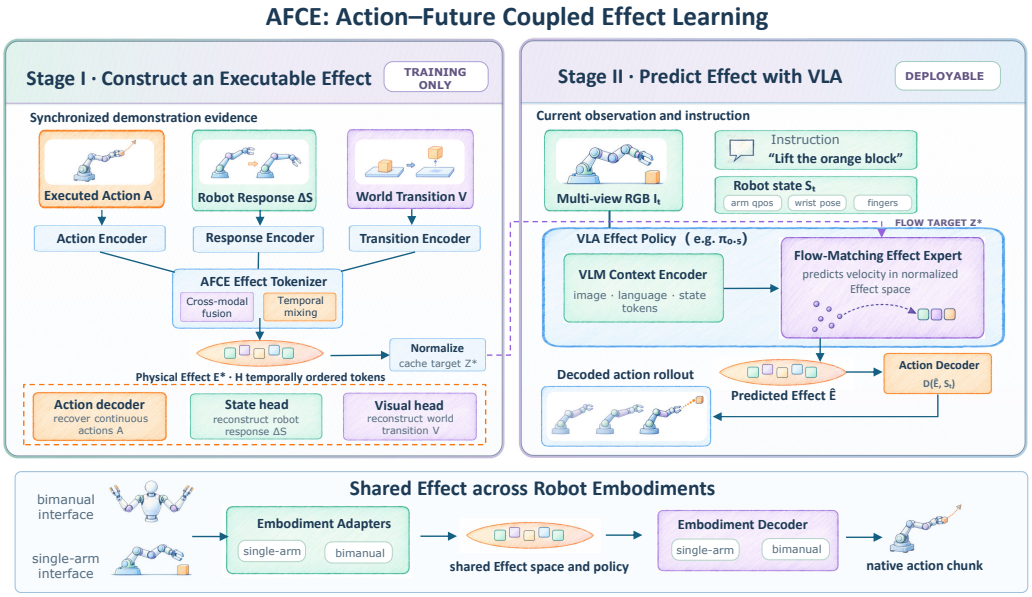



Figure 2: **Overview of Action–Future Coupled Effect Learning (AFCE).** Stage I fuses demonstrated actions $\mathbf{A}_t$, robot response $\Delta\mathbf{S}_t$, and visual transitions $\mathbf{V}_t$ into ordered Effect tokens $\mathbf{E}_t^*$, supervised by action, state, and visual reconstruction. Stage II predicts the normalized target $\mathbf{Z}_t^*$ from current images, instruction, and robot state, then decodes an action chunk. The lower panel shows the shared Effect space with embodiment-specific interfaces. The dashed path denotes training targets; future evidence is not used at deployment.

Here $p_\theta$ is implemented by a conditional flow-matching VLA and $\mathcal{N}$ is the fixed Effect normalization. Future observations enter Stage I target construction only; the deployed path uses current images, instruction, and robot state.

### 3.1 Stage I: Learning an Executable, Future-Grounded Effect

**Synchronized evidence.** The three evidence sources describe distinct aspects of the same interaction. The action encoder uses group-specific projections for position, rotation, and finger commands before combining them into step-level features. The response encoder represents realized palm translation, relative orientation, fingertip displacement, and finger-joint change in the reference frame of the prediction window. Its features are computed from observed states and forward kinematics and aligned with future action steps; both arms are retained for dual-arm demonstrations. Separating commands from realized motion allows the Effect to describe the consequences of execution rather than merely re-encode the commands.

For world transitions, we sample frozen DINOv3 features at sparse times. A localized visual token records appearance-feature change, image-plane displacement, spatial location, and temporal extent. The evidence may contain robot as well as object motion and does not require perfect object segmentation. Temporal metadata associates a transition with the action steps covered by its observation interval; a change spanning several steps is not treated as an independent observation at every control instant. The exact feature construction and temporal assignment are given in the appendix.

**Fusion before compression.** The encoders map their inputs to a common width, producing action and response features $\mathbf{z}_k^A$ and $\mathbf{z}_k^S$, together with visual features $\mathcal{Z}_k^V$ associated with step $k$. Modality and temporal embeddings identify their sources and positions. A learned step query first fuses the available evidence through cross-attention; a temporal Transformer then exchanges information across the full window:

$$\begin{aligned}\mathbf{u}_k &= F_{\text{fuse}}\left(\mathbf{q}_k; \mathbf{z}_k^A, \mathbf{z}_k^S, \mathcal{Z}_k^V\right), \\ \mathbf{E}_t^* &= F_{\text{temp}}([\mathbf{u}_0, \ldots, \mathbf{u}_{H-1}]).\end{aligned} \tag{3}$$

Thus, action, response, and world-transition evidence interact *before* the final Effect tokens are formed. Each token keeps its temporal position while incorporating context from other steps. Because $T_\phi$ processes complete training windows, its temporal attention need not be causal; this does not expose future evidence to the deployed policy.

**Executable-bottleneck supervision.** A state-conditioned decoder reconstructs the demonstrated actions:

$$\widetilde{\mathbf{A}}_t = D_\psi(\mathbf{E}_t^*, \mathbf{s}_t). \tag{4}$$

Auxiliary state and visual heads reconstruct the realized robot response and sparse visual transitions. The visual head receives the spatial and temporal query metadata specifying *where and when* to decode, but not the target feature change. We minimize

$$\mathcal{L}_{\mathrm{I}} = \mathcal{L}_A + \lambda_S(u)\mathcal{L}_S + \lambda_V(u)\mathcal{L}_V + \lambda_R\mathcal{L}_{\mathrm{rot}}, \tag{5}$$

with the grouped *Huber* action-reconstruction loss

$$\mathcal{L}_A = \frac{2\mathcal{H}_p + \mathcal{H}_r + \mathcal{H}_f}{3}. \tag{6}$$

Here $p$, $r$, and $f$ denote position, rotation, and finger groups, and $\mathcal{H}_g$ is the corresponding normalized Huber error. Each group is averaged over valid coordinates and time steps so its dimensionality does not set its effective weight. The denominator remains three despite the position weight of two. The state and visual losses likewise average errors over valid, normalized targets. $\mathcal{L}_{\mathrm{rot}}$ is an SO(3) chordal loss that regularizes decoded orientations in addition to rotation-coordinate supervision. Action reconstruction preserves executable detail, whereas state and visual reconstruction encourage the Effect to retain realized physical changes.

Figure 3 shows a hammer–nail example in which decoded robot motion accompanies visual scene change. The decoder reads the Effect and current robot state, rather than the future frame, to produce the illustrated motion. This example makes the link between the learned representation and executable action concrete, while the accompanying scene images show the physical transition used to construct its training target. The visualization is qualitative: successful reconstruction here does not establish that every evidence source contributes to the Effect, or that the policy can predict it accurately. We examine those questions through modality interventions, GT-Effect replay, and closed-loop execution.

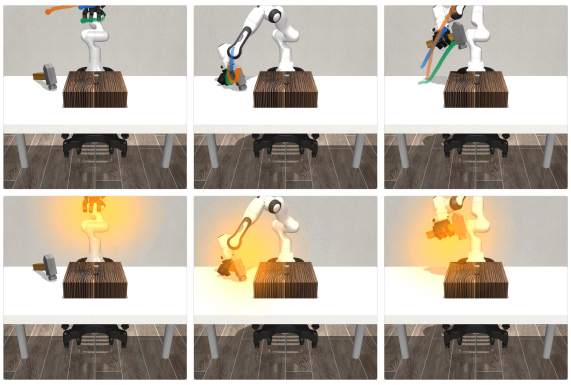


Figure 3: **Action and scene changes represented by the Effect.** Three successive hammer-nail frames. Top: decoded robot motion; bottom: accompanying scene changes.

**Fixed targets for Stage II.** After Stage I, we freeze $T_\phi$ and estimate channel-wise mean $\boldsymbol{\mu}_E$ and standard deviation $\boldsymbol{\sigma}_E$ on all Stage-I training demonstrations. The normalized target is

$$\mathbf{Z}_t^* = \mathcal{N}(\mathbf{E}_t^*) = \frac{\mathbf{E}_t^* - \boldsymbol{\mu}_E}{\max(\boldsymbol{\sigma}_E, \sigma_{\min})}, \tag{7}$$

where the maximum is elementwise and $\sigma_{\min}$ is a fixed standard-deviation floor. Statistics are shared across temporal positions and remain fixed throughout Stage II. We cache the Stage-I Effect outputs for policy training and apply this fixed normalization to form targets. The future-evidence encoders and auxiliary heads are not used at deployment.

### 3.2 Stage II: Predicting and Executing Effects

**Conditional Effect flow matching.** We adapt $\pi_{0.5}$ to predict normalized Effects: its action input and output projections are replaced with projections of width $D_E$, while its visual-language-state conditioning and flow-matching backbone are retained. Let $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ have the shape of $\mathbf{Z}_t^*$ and

let $\tau \in [0, 1]$ be flow time, distinct from environment time $t$. We interpolate from the clean target at $\tau = 0$ to noise at $\tau = 1$:

$$\mathbf{Z}_{t,\tau} = (1 - \tau)\mathbf{Z}_t^* + \tau\boldsymbol{\epsilon}, \qquad \mathbf{v}^* = \boldsymbol{\epsilon} - \mathbf{Z}_t^*. \tag{8}$$

The conditional velocity field is trained with

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t,\tau,\boldsymbol{\epsilon}}\left[\frac{1}{HD_E}\left\|\mathbf{v}_\theta(\mathbf{Z}_{t,\tau}, \tau; \mathbf{o}_t) - \mathbf{v}^*\right\|_F^2\right]. \tag{9}$$

At inference, integrating the learned flow from noise toward $\tau = 0$ gives a normalized predicted Effect.

**Action-grounded prediction.** Distance in Effect space need not reflect the action error caused by decoding. We therefore supervise a clean-endpoint estimate and its decoded action chunk:

$$\begin{aligned}\widehat{\mathbf{Z}}_{t,0}^{(\tau)} &= \mathbf{Z}_{t,\tau} - \tau\mathbf{v}_\theta(\mathbf{Z}_{t,\tau}, \tau; \mathbf{o}_t),\\ \widehat{\mathbf{A}}_t^{(\tau)} &= D_\psi\Big(\mathcal{N}^{-1}(\widehat{\mathbf{Z}}_{t,0}^{(\tau)}), \mathbf{s}_t\Big).\end{aligned} \tag{10}$$

This estimate follows from Equation 8 and avoids differentiating through a complete sampling trajectory. The ground-truth-action auxiliary loss $\mathcal{L}_{\mathrm{GT}}$ compares $\widehat{\mathbf{A}}_t^{(\tau)}$ with the *demonstrated* action after fixed action standardization. It averages position, rotation, and finger Huber errors with equal group weights. Unlike Equation 6, it does not apply the Stage-I position multiplier or SO(3) term.

The decoder is initialized from Stage I and updated jointly with the policy at a reduced learning rate. Thus, it adapts to predicted Effects, and action-loss gradients also reach the policy. The same decoder is used during supervision and deployment.

We additionally match changes in finger commands across consecutive steps. Let $f_{b,k,j}$ and $\widehat{f}_{b,k,j}^{(\tau)}$ denote standardized demonstrated and decoded finger commands for batch element $b$, step $k$, and finger joint $j$. Define $\Delta f_{b,k,j} = f_{b,k+1,j} - f_{b,k,j}$, with $\Delta\widehat{f}_{b,k,j}^{(\tau)}$ defined analogously. The finger-difference loss is

$$\mathcal{L}_{\Delta f} = \operatorname*{mean}_{(b,k,j)\ \mathrm{valid}} \rho_1\left(\frac{\Delta\widehat{f}_{b,k,j}^{(\tau)} - \Delta f_{b,k,j}}{\bar{s}_j}\right), \tag{11}$$

where $\rho_1$ is the Huber loss with threshold one. The scale $\bar{s}_j$ is the stopped-gradient, batch-level RMS of demonstrated adjacent differences for joint $j$, subject to a fixed minimum. Both steps of an adjacent pair must satisfy the Effect-valid mask. This loss supervises finger opening and closing without treating joints with different difference magnitudes as equally scaled.

**Joint optimization and closed-loop execution.** Stage II minimizes

$$\mathcal{L}_{\mathrm{II}} = \mathcal{L}_{\mathrm{FM}} + r(u)\,\mathbb{E}_\tau\left[g(\tau)\left(\lambda_{\mathrm{GT}}\mathcal{L}_{\mathrm{GT}} + \lambda_{\Delta f}\mathcal{L}_{\Delta f}\right)\right], \tag{12}$$

where $u$ is the optimization step, $r(u) \in [0, 1]$ warms up both action losses, and $g(\tau)$ selects flow times at which the clean-endpoint estimate is supervised. Flow matching remains active over the full training distribution. The gate also accounts for the probability mass of its selected flow-time interval. The ground-truth-action weight is calibrated with reference to its gradient scale relative to flow matching. Flow-time sampling, gating, coefficients, and optimization schedules are provided in Sec. D.

At deployment, the VLA predicts $\widehat{\mathbf{Z}}_t$ from the current observation, $\mathcal{N}^{-1}$ recovers the Effect, and the jointly trained decoder produces an action chunk. We execute a prefix, observe again, and replan using the same receding-horizon protocol as the direct-action baseline. The deployed system retains the VLA, fixed normalization statistics, the required state projection, and the action decoder.

### 3.3 Cross-Embodiment Factorization

Single- and dual-arm demonstrations use different native state and action dimensions. AFCE uses separate input and output adapters while keeping a common Effect shape. For embodiment or control configuration $e$,

$$\mathbf{h}^{A,e} = f_A^e(\mathbf{A}^e), \qquad \mathbf{h}^{S,e} = f_S^e(\Delta\mathbf{S}^e), \qquad \widehat{\mathbf{A}}^e = D_\psi^e\Big(\widehat{\mathbf{E}}, \mathbf{s}_t^e\Big). \tag{13}$$

Action and response adapters map native coordinates to a common feature width. State projections and action-decoding interfaces are configuration-specific, whereas the visual encoder, cross-modal fusion, temporal Effect modeling, and Effect-prediction backbone are shared. The matching decoder $D^e_\psi$ is used for both Stage II supervision and deployment.

For configuration-specific demonstration distributions $\mathcal{D}^e_j$, mixed training at either stage $j \in \{\mathrm{I}, \mathrm{II}\}$ can be written as

$$\mathcal{L}^{\mathrm{mix}}_j = \sum_{e\in\mathcal{E}} \omega^{(j)}_e \mathbb{E}_{\xi\sim\mathcal{D}^e_j}\left[\mathcal{L}^e_j(\xi)\right], \qquad \omega^{(j)}_e \geq 0, \quad \sum_{e\in\mathcal{E}} \omega^{(j)}_e = 1. \tag{14}$$

The weights describe the mixture induced by the training sampler. In Stage II, sampling is proportional to eligible frame counts rather than assigning equal weight to each task. Reconstruction losses are averaged over valid coordinates within each configuration, preventing larger native action spaces from receiving greater weight solely due to their dimensionality. This factorization enables heterogeneous demonstrations to train a shared Effect representation without a common native action space. It does not require different robots to produce identical Effect codes or follow identical trajectories for similar outcomes; embodiment-invariant alignment would require separate evidence.

# 4 Experiments

## 4.1 Experimental Setup

**Benchmark and compared methods.** We evaluate on DexJoCo (Wang et al., 2026), which comprises six single-arm and five bimanual dexterous manipulation tasks. All reported evaluations use `rand-obj`, with randomized object placement and table height. We compare our method with UVT (Feng & Jain, 2026) and the officially released DexJoCo $\pi_{0.5}$ multi-task checkpoint, denoted as $\pi_{0.5}$ in Table 1. Inputs comprise scene and wrist images, the language instruction, and proprioceptive state.

**Evaluation protocol.** Each method is evaluated for 50 episodes per task under each of three environment seeds (0, 1, and 2), totaling 1,650 episodes per method. We report mean success rate and sample standard deviation across these *evaluation* seeds, using one trained checkpoint per method. These statistics describe variation across evaluation seeds, not independent training runs. For each group, we first average task success rates within each seed and then compute the mean and sample standard deviation across seeds. The overall score averages all 11 tasks equally.

Table 1: Success rates (%) on DexJoCo under `rand-obj`.

| | Single-arm | | | | | | Bimanual | | | | | Mean | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **PT** | **CM** | **FG** | **HN** | **PB** | **WP** | **UI** | **HA** | **AS** | **MW** | **PH** | **SA** | **BA** | **All** |
| UVT | 1.33 ±1.15 | **80.00** ±2.00 | **64.67** ±5.03 | 22.00 ±6.93 | 60.00 ±6.00 | **92.67** ±2.31 | 0.00 ±0.00 | 0.67 ±1.15 | 0.00 ±0.00 | 41.33 ±11.02 | 30.67 ±4.16 | 53.44 ±2.22 | 14.53 ±3.06 | 35.76 ±0.56 |
| $\pi_{0.5}$ | **43.33** ±8.08 | 60.00 ±2.00 | 38.00 ±5.29 | 84.67 ±5.77 | 67.33 ±1.15 | 88.67 ±1.15 | 2.00 ±0.00 | 5.33 ±4.16 | 1.33 ±2.31 | 32.00 ±8.72 | 8.00 ±3.46 | **63.67** ±1.86 | 9.73 ±2.41 | 39.15 ±2.11 |
| **Ours** | 5.33 ±2.31 | 54.67 ±7.57 | 62.00 ±6.93 | **86.67** ±5.77 | **80.67** ±9.02 | 84.67 ±9.02 | **7.33** ±1.15 | **29.33** ±1.15 | **9.33** ±3.06 | **81.33** ±6.43 | **59.33** ±4.16 | 62.33 ±1.45 | **37.33** ±2.66 | **50.97** ±1.65 |

Mean ± sample standard deviation across three environment seeds; 50 episodes per task per seed. Bold denotes the highest mean among reported methods.
PT: Pinch Tongs; CM: Click Mouse; FG: Fold Glasses; HN: Hammer Nail; PB: Pick Bucket; WP: Water Plant; UI: Unlock iPad; HA: Hanoi; AS: Assembly; MW: Microwave; PH: Photograph. SA/BA/All: single-arm, bimanual, and overall task means.

## 4.2 Main Results

**Overall performance.** Ours achieves $50.97 \pm 1.65\%$ (841/1,650 episodes), exceeding $\pi_{0.5}$ by 11.82 percentage points ($39.15 \pm 2.11\%$) and UVT by 15.21 points ($35.76 \pm 0.56\%$). It obtains the highest mean on seven tasks.

**Bimanual manipulation.** Ours leads on all five bimanual tasks, reaching a mean of $37.33\%$, compared with $9.73\%$ for $\pi_{0.5}$ and $14.53\%$ for UVT. The largest gains over $\pi_{0.5}$ occur on Microwave ($81.33\%$ versus $32.00\%$) and Photograph ($59.33\%$ versus $8.00\%$). Success on Hanoi also rises from

5.33% to 29.33%. Ours improves on Assembly and Unlock iPad, although their success rates remain low at 9.33% and 7.33%, respectively.

**Single-arm manipulation.** Ours achieves a single-arm mean of 62.33%, slightly below $\pi_{0.5}$ (63.67%) but above UVT (53.44%). It leads on Hammer Nail and Pick Bucket and improves over $\pi_{0.5}$ on Fold Glasses (62.00% versus 38.00%), although UVT performs best on the latter task. Pinch Tongs is the largest regression: Ours reaches 5.33%, versus 43.33% for $\pi_{0.5}$. The seed-0 precision-task failure rate in Figure 4 likewise rises from 51.3% to 58.7%. Thus, the overall gain does not extend uniformly to fine-grained single-arm control.

**Source of the aggregate gain.** With equal numbers of evaluation episodes per task, the 27.60-point bimanual gain over $\pi_{0.5}$ contributes approximately 12.55 points to the overall mean; the 1.34-point single-arm deficit offsets approximately 0.73 points. The resulting net gain is 11.82 points.

Figure 4 groups the tasks into Tool use (Hammer Nail, Water Plant), Grasping (Pick Bucket), Precision (Pinch Tongs, Click Mouse, Fold Glasses), Bimanual (Microwave, Photograph), Coordination (Hanoi, Assembly), and Sequential (Unlock iPad). The groups are disjoint; in particular, the figure's Bimanual group contains two tasks, whereas the BA column of Table 1 averages all five bimanual tasks.

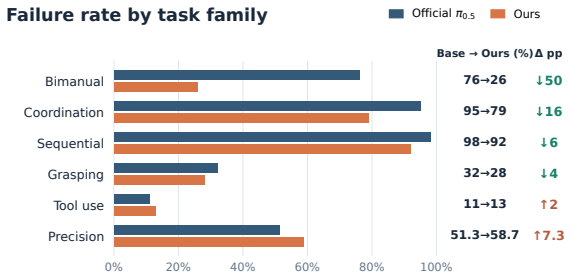



Figure 4: Failure rates by task family for evaluation seed 0. Arrows show changes from $\pi_{0.5}$ to Ours in percentage points.

On seed 0, Ours reduces the failure rate from 76% to 26% in Bimanual and from 95% to 79% in Coordination. Sequential improves from 98% to 92%, but Unlock iPad remains difficult. Failure rates instead rise in Precision (51.3% to 58.7%) and Tool use (11% to 13%). This analysis localizes gains and regressions of the complete policies; the ablations examine the contribution of Effect-construction inputs.

### 4.3 Effect Dimensionality Sensitivity

We train $30 \times d$ Effects with $d \in \{256, 128, 64\}$ for 30,000 steps under identical settings. Table 2 reports decoding errors and GT-Effect replay on the same selected set of 110 training demonstrations, with 10 demonstrations per task. This diagnostic evaluates the Effect and decoder without policy-side Effect prediction and is reported separately from closed-loop policy success.

Table 2: Effect dimensionality sensitivity.

| | Error (mm) ↓ | | | GT-Effect replay ↑ | | |
|---|---|---|---|---|---|---|
| $d$ | Pos. | Bi-pos. | FK | SA | BA | All |
| **256** | **7.794** | **9.507** | **10.519** | 20/60 | **5/50** | **25/110** |
| 128 | 8.965 | 11.455 | 11.716 | **21/60** | 2/50 | 23/110 |
| 64 | 11.585 | 15.272 | 14.980 | 18/60 | 0/50 | 18/110 |

Pos.: overall position error; Bi-pos.: bimanual position error; FK: fingertip forward-kinematics position error. SA/BA/All: successful single-arm, bimanual, and total replay episodes. Bold indicates the best value in each column.

Reducing $d$ from 256 to 64 increases all three decoding errors and lowers total GT-Effect replay success from 25/110 to 18/110. The trend differs slightly by task group: single-arm replay rises from 20/60 to 21/60 at $d = 128$, but bimanual replay falls from 5/50 to 2/50 and reaches 0/50 at $d = 64$. We therefore retain $d = 256$, which gives the lowest measured reconstruction errors and highest total replay success among these variants. The modest absolute replay rate also cautions against treating representation reconstruction alone as evidence of reliable execution.

### 4.4 EFFECT-CONSTRUCTION ABLATION

Table 3: Effect-construction input ablation on DexJoCo under `rand-obj`. Success rates are reported in %.

| Variant | Effect inputs | | | Single-arm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $S$ | $V$ | $A$ | **PT** | **CM** | **FG** | **HN** | **PB** | **WP** | **SA** |
| $A$ only | – | – | ✓ | 1.33 ±1.15 | 65.33 ±3.06 | 54.67 ±10.07 | 68.00 ±2.00 | 69.33 ±6.11 | 84.67 ±2.31 | 57.22 ±0.38 |
| w/o $S$ | – | ✓ | ✓ | 5.33 ±5.03 | 48.00 ±5.29 | 51.33 ±4.62 | 76.00 ±2.00 | 84.67 ±7.02 | 26.67 ±5.03 | 48.67 ±1.53 |
| w/o $V$ | ✓ | – | ✓ | 14.00 ±5.29 | 57.33 ±3.06 | 54.67 ±6.43 | 78.67 ±8.08 | 76.67 ±9.02 | 52.67 ±12.06 | 55.67 ±3.06 |
| **AFCE** | ✓ | ✓ | ✓ | 5.33 ±2.31 | 54.67 ±7.57 | 62.00 ±6.93 | 86.67 ±5.77 | 80.67 ±9.02 | 84.67 ±9.02 | **62.33** ±1.45 |

| Variant | Effect inputs | | | Bimanual | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $S$ | $V$ | $A$ | **UI** | **HA** | **AS** | **MW** | **PH** | **BA** |
| $A$ only | – | – | ✓ | 2.67 ±1.15 | 28.67 ±8.33 | 4.00 ±5.29 | 76.00 ±2.00 | 43.33 ±7.57 | 30.93 ±2.81 |
| w/o $S$ | – | ✓ | ✓ | 3.33 ±1.15 | 11.33 ±4.16 | 1.33 ±1.15 | 57.33 ±9.87 | 54.00 ±6.93 | 25.47 ±2.66 |
| w/o $V$ | ✓ | – | ✓ | 1.33 ±1.15 | 34.00 ±3.46 | 2.00 ±0.00 | 61.33 ±5.03 | 9.33 ±4.62 | 21.60 ±1.20 |
| **AFCE** | ✓ | ✓ | ✓ | 7.33 ±1.15 | 29.33 ±1.15 | 9.33 ±3.06 | 81.33 ±6.43 | 59.33 ±4.16 | **37.33** ±2.66 |

A checkmark indicates evidence used to construct the Effect target; – indicates its removal. $S$: realized robot response; $V$: future world transition; $A$: executed actions. SA and BA denote the single-arm and bimanual task means. Values are mean ± sample standard deviation across three environment seeds, with 50 episodes per task per seed. Task abbreviations follow Table 1.

Table 3 varies the evidence supplied to the Effect tokenizer while retaining executed actions as an input. With only $A$ as input, the policy achieves $57.22\%$ success on single-arm tasks and $30.93\%$ on bimanual tasks. Adding $V$ alone yields $48.67\%$ and $25.47\%$, while adding $S$ alone yields $55.67\%$ and $21.60\%$. The complete $A + S + V$ configuration achieves the highest group means, $62.33\%$ and $37.33\%$, exceeding the $A$-only variant by $5.11$ and $6.40$ percentage points.

At the task level, full AFCE improves Hammer Nail from $68.00\%$ to $86.67\%$ and Photograph from $43.33\%$ to $59.33\%$ relative to $A$ only, but reduces Click Mouse from $65.33\%$ to $54.67\%$. The nonmonotonic results support the complete configuration at the group level without establishing an independent benefit from each added input; closed-loop success alone cannot distinguish errors in Effect prediction, fusion, and action decoding.

## 5 LIMITATIONS

AFCE requires temporally aligned actions, robot states, and future images to construct training targets. Visual features can miss subtle contacts or be affected by occlusion and viewpoint changes. A fixed-size Effect must also preserve control detail while remaining predictable from current observations; the Pinch Tongs result illustrates an unresolved precision-control limitation. Prediction errors can further degrade action readout, and offline target construction increases training cost. Finally, shared parameters across single- and bimanual interfaces do not establish embodiment-invariant representations.

## 6 CONCLUSION

We introduced AFCE, which constructs a temporal Effect from demonstrated actions, realized robot responses, and sparse visual transitions, then predicts and decodes it for closed-loop control. On DexJoCo under `rand-obj`, AFCE achieves 50.97% overall success, compared with 39.15% for $\pi_{0.5}$ and 35.76% for the evaluated UVT baseline. The gains are strongest on bimanual tasks, while precision-sensitive single-arm control remains challenging. AFCE uses future physical evidence for supervision while requiring only current observations at inference time.

## AI USE STATEMENT

Generative AI tools were used to assist with discussing experimental design, writing and editing research code, organizing and interpreting experimental results, preparing figure drafts, and improving the clarity of the manuscript. The authors made the final methodological decisions, reviewed and validated AI-assisted code and figures, conducted the experiments, and checked all reported results against experiment records. The authors take full responsibility for the content of this paper.

## ETHICS STATEMENT

This work studies robotic manipulation in simulation using existing DexJoCo demonstrations. We conducted no new human-subject study and collected no personal data. We use the benchmark, datasets, and pretrained models under their applicable licenses. The resulting policies have not been validated for physical deployment; use on real robots would require further safety testing, appropriate control limits, and human supervision.

## REPRODUCIBILITY STATEMENT

We describe the AFCE architecture, training objectives, data processing, and key hyperparameters for both training stages. We also specify the DexJoCo tasks, demonstration data, training and evaluation seeds, evaluated checkpoints, rollout counts, and task-specific success criteria. Task-level results and additional analyses are provided to support interpretation and reproduction of the reported findings.

## REFERENCES

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. $\pi_0$: 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 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 Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025.

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.

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

Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, Jianyu Chen, and Jiang Bian. villa-x: Enhancing latent action modeling in vision-language-action models. In *International Conference on Learning Representations*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Zhenyang Feng and Unnat Jain. Unified visuomotor targets: Supervising vlas beyond physical actions, 2026. URL https://arxiv.org/abs/2608.03563.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. 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. In *Proceedings of the Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025.

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

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*, 2026a.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-H0.7: A latent world-action model from egocentric videos. *arXiv preprint arXiv:2605.00078*, 2026b.

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

Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, Lue Fan, Hongsheng Li, Zhaoxiang Zhang, and Tieniu Tan. DexJoCo: A benchmark and toolkit for task-oriented dexterous manipulation on MuJoCo. *arXiv preprint arXiv:2605.16257*, 2026.

Youpeng Wen, Junfan Lin, Yi Zhu, Jianhua Han, Hang Xu, Shen Zhao, and Xiaodan Liang. Vidman: Exploiting implicit dynamics from video diffusion model for effective robot manipulation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

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.

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

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune 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. *arXiv preprint arXiv:2410.11758*, 2024.

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

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. *arXiv preprint arXiv:2507.04447*, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023.

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

# A Method and Objective Details

AFCE separates effect construction from effect prediction. Given a demonstration window beginning at time $t$, the first stage encodes the executed actions $A_{t:t+H-1}$, realized robot responses $\Delta S_{t:t+H-1}$, and future visual transitions $V_t$ into an Effect target $E_t^*$. The second stage predicts this target from the current image, instruction, and robot state, and decodes the predicted Effect into an executable action chunk. Demonstration futures are used to construct training targets but are not inputs to the deployed policy.

## A.1 Effect Evidence and Architecture

The action branch represents the demonstrated control sequence. The response branch describes realized changes in palm pose, fingertip positions, and finger joints relative to the beginning of the window. The visual branch extracts sparse transition descriptors from future observations using a frozen visual encoder. These descriptors may include robot motion; the visual stream is not restricted to object-only changes.

Each branch produces temporally aligned features, which are projected to a shared width and augmented with modality embeddings. At each future step, a learned query attends to the action, response, and visual features through QueryFusion. A temporal Transformer then exchanges information across the window to produce

$$E_t^* = T_\phi(A_{t:t+H-1}, \Delta S_{t:t+H-1}, V_t) \in \mathbb{R}^{H \times d_E}. \tag{A.1}$$

A state-conditioned decoder $D_\psi^e$ maps the Effect to an action sequence in the native coordinates of embodiment $e$. Separate output interfaces accommodate different action dimensions while the Effect representation has a common shape.

## A.2 Learning an Executable Effect

The tokenizer is trained with action reconstruction and auxiliary response and visual reconstruction. Let $\mathcal{H}_p$, $\mathcal{H}_r$, and $\mathcal{H}_f$ denote grouped Huber errors for standardized position, rotation, and finger actions, averaged over valid steps and coordinates. The action loss is

$$\mathcal{L}_A = \frac{w_p \mathcal{H}_p + \mathcal{H}_r + \mathcal{H}_f}{3}, \tag{A.2}$$

where $w_p$ controls the emphasis on position reconstruction. The denominator remains three when $w_p \neq 1$. We additionally measure the discrepancy between decoded and demonstrated orientations by the SO(3) chordal loss

$$\mathcal{L}_{\mathrm{SO}(3)} = \mathbb{E}\left[\frac{\|R(\hat{A}) - R(A)\|_F^2}{6}\right]. \tag{A.3}$$

The response and visual heads reconstruct their respective normalized targets on valid entries, yielding losses $\mathcal{L}_S$ and $\mathcal{L}_V$. The complete first-stage objective is

$$\mathcal{L}_{\mathrm{I}} = \mathcal{L}_A + \lambda_S(u)\mathcal{L}_S + \lambda_V(u)\mathcal{L}_V + \lambda_R \mathcal{L}_{\mathrm{SO}(3)}, \tag{A.4}$$

where $u$ is the optimizer step. The auxiliary weights are gradually increased during tokenizer training. Their schedules and the remaining numerical settings are given in Sec. D.

After training, we encode the demonstration windows into fixed Effect targets. Channel-wise statistics fitted on all tokenizer training demonstrations define the normalized policy target

$$Z_t^* = \frac{E_t^* - \mu_E}{\sigma_E^{\mathrm{safe}}}, \tag{A.5}$$

where $\sigma_E^{\mathrm{safe}}$ includes a minimum standard-deviation threshold. The tokenizer and normalization statistics remain fixed in the second stage.

### A.3 Effect Prediction and Action Readout

The AFCE policy is initialized from $\pi_{0.5}$, with its action projections adapted to the Effect dimension. Conditioned on the current observation and robot state, it learns a velocity field in normalized Effect space. For $\epsilon \sim \mathcal{N}(0, I)$ and flow time $\tau$, we construct

$$Z_{t,\tau} = (1-\tau)Z_t^* + \tau\epsilon, \qquad v^* = \epsilon - Z_t^*. \tag{A.6}$$

The flow-matching objective averages squared velocity error over valid Effect coordinates:

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{\tau,\epsilon}\left[\mathrm{mean}_{k,j}\left(v_{\theta,k,j} - v^*_{k,j}\right)^2\right]. \tag{A.7}$$

To connect Effect prediction to executable control during training, we form the single-step clean estimate

$$\hat{Z}^{(\tau)}_{t,0} = Z_{t,\tau} - \tau v_\theta(Z_{t,\tau}, \tau; X_t), \qquad \hat{A}^{(\tau)}_t = D^e_\psi\left(\mathcal{N}_E^{-1}(\hat{Z}^{(\tau)}_{t,0}), S_t\right), \tag{A.8}$$

where $X_t$ is the current image–language–state context and $\mathcal{N}_E^{-1}$ reverses Effect standardization. The ground-truth-action auxiliary loss $\mathcal{L}_{\mathrm{GT}}$ compares $\hat{A}^{(\tau)}_t$ with the standardized demonstrated action. It averages position, rotation, and finger Huber errors with equal group weights; the position multiplier and SO(3) term from Stage I are not applied here.

A second auxiliary loss preserves finger motion within the decoded chunk. For consecutive valid steps, define $\Delta f_{t+k} = f_{t+k+1} - f_{t+k}$ and the corresponding decoded difference $\Delta \hat{f}_{t+k}$. Each joint's difference error is scaled by the stopped-gradient RMS of its demonstrated differences, subject to a minimum scale, before applying a Huber loss. Averaging over valid adjacent pairs and finger joints gives $\mathcal{L}_{\Delta f}$.

Both action losses update the policy and the same embodiment decoder used at inference. They share a flow-time gate $g(\tau)$ and a training ramp $r(u)$:

$$\mathcal{L}_{\mathrm{II}} = \mathcal{L}_{\mathrm{FM}} + r(u)g(\tau)\left(\lambda_{\mathrm{GT}}\mathcal{L}_{\mathrm{GT}} + \lambda_{\Delta f}\mathcal{L}_{\Delta f}\right). \tag{A.9}$$

The gate restricts action supervision to a designated flow-time interval and accounts for its probability mass under the sampling distribution. The exact interval, weights, ramp, and optimizer settings are reported in Sec. D. At inference, the policy generates a normalized Effect, which is denormalized and passed through this jointly trained decoder to obtain a native action chunk. The future-evidence tokenizer is not run online.

## B Datasets and Evaluation Protocols

We evaluate AFCE on all 11 DexJoCo tasks (Wang et al., 2026), comprising six single-arm and five bimanual Panda–Allegro manipulation tasks. These tasks involve tool use, articulated finger motions, ordered interactions, and bimanual coordination. Table B.1 summarizes their functional goals. Success is determined by each task's simulator-defined completion condition.

We use all 100 recorded episodes per task, totaling 1,100 episodes and 523,763 retained frames at 30 FPS, to train both the Stage-I Effect tokenizer and the Stage-II policy. Effect targets are constructed offline from demonstration actions, robot responses, and future visual transitions. During rollout, the policy receives only the current image observation, language instruction, and robot proprioception; future evidence and privileged simulator object states are unavailable. Each arm's action contains end-effector position (3), rotation vector (3), and finger joints (16), giving 22 action dimensions for single-arm tasks and 44 for bimanual tasks. The corresponding proprioceptive state has 23 or 46 dimensions. The visual-transition pipeline uses the ego view for bimanual tasks, `ego_right` for Click Mouse, and the front view for the remaining single-arm tasks.

Closed-loop evaluation uses the DexJoCo `rand-obj` setting, which randomizes object placement and table height. The results in our main table aggregate 150 rollouts per task, giving 1,650 rollouts per method across all 11 tasks. At each policy query, ten flow denoising steps produce a 30-step Effect chunk; the jointly trained decoder converts it into actions for execution and subsequent replanning. An episode succeeds when the benchmark's task-specific completion condition is met. If $n_i$ denotes the number of successful rollouts on task $i$, we report $\mathrm{SR}_i = 100n_i/150$ and $\mathrm{SR}_{\mathrm{all}} = 100\sum_{i=1}^{11} n_i/1650$.

Table B.1: Functional goals of the 11 DexJoCo tasks.

| Task | Objective |
|---|---|
| *Single arm* | |
| Click Mouse | Move the mouse onto its pad and click. |
| Fold Glasses | Fold the glasses and place them in the case. |
| Hammer Nail | Drive a nail into a board with a hammer. |
| Pick Bucket | Place food in the bucket and lift it. |
| Pinch Tongs | Grasp the tongs and open and close them three times. |
| Water Plant | Use a watering can to water the plant. |
| *Bimanual* | |
| Assembly | Hold a tray and insert a peg into its hole. |
| Hanoi | Complete the final two moves of a three-level Tower of Hanoi. |
| Microwave | Place food inside, close the door, and press start. |
| Photograph | Aim a camera and press its shutter. |
| Unlock iPad | Hold the device and enter the password 123. |

The single-arm and bimanual aggregates comprise 900 and 750 rollouts, respectively. AFCE achieves 841/1,650 overall successes (50.97%), including 561/900 (62.33%) on single-arm tasks and 280/750 (37.33%) on bimanual tasks. Reconstruction and GT-Effect replay on the demonstrations are reported as training-data diagnostics, separately from closed-loop policy success.

# C Additional Experimental Results

We provide complementary diagnostics of the Effect representation, its action readout, and the sparse visual evidence used to construct it. Unless stated otherwise, the analyses use the frozen AFCE Effect tokenizer at update 30,000 and the policy checkpoint at update 59,999. The 110 diagnostic trajectories (ten per task) belong to the demonstrations used to train both the Stage-I tokenizer and the Stage-II policy. These analyses are in-sample diagnostics, not held-out validation or closed-loop policy evaluations.

For the action-readout diagnostic, $D_0$ denotes the frozen Stage-I decoder, whereas $D_1$ is the jointly adapted Stage-II decoder used at deployment.

**Effect-space structure and prediction.** Figure C.1 displays a joint PCA–t-SNE projection of 1,100 GT-E windows and 1,100 corresponding policy-predicted windows. Each window contains $30 \times 256$ effect values. Task and trajectory-progress colors reveal visible organization in the GT-E points; the fourth panel reports paired prediction error in the original normalized effect space rather than in the two-dimensional projection. Overlap of projected GT-E and predicted effects alone does not establish accurate action readout: the matched position error rises from 8.30 mm with GT-E/$D_1$ to 36.91 mm with predicted-E/$D_1$ (Figure C.3). Figure C.2 shows that the apparent two-dimensional geometry also depends on t-SNE settings.

**Projection sensitivity and physical readout error.** Figure C.2 compares three prespecified t-SNE settings for the same GT-E windows and fixed PCA representation. The changing layout warns against reading inter-cluster distances as quantitative physical or semantic distances. Figure C.3 compares action readout on matched demonstration windows: GT-E/$D_0$ and GT-E/$D_1$ produce position errors of 8.00 and 8.30 mm, respectively, while predicted-E/$D_1$ yields 36.91 mm. Rotation and finger errors likewise rise for predicted-E/$D_1$ (7.68 degrees and 2.97 degrees). These are errors against demonstration actions, not closed-loop task success rates.

**Qualitative task-progress comparisons.** Figure C.4 compares illustrative AFCE and baseline rollouts at four relative progress points. In Fold Glasses, the baseline contacts the frame but leaves the glasses on the table, whereas AFCE folds and places them in the tray. In Hanoi, AFCE maintains a stable grasp and aligns the piece with the target peg, while the baseline ends with a tilted placement. AFCE also completes the ordered interactions in Microwave, progressing from the door to the controls,

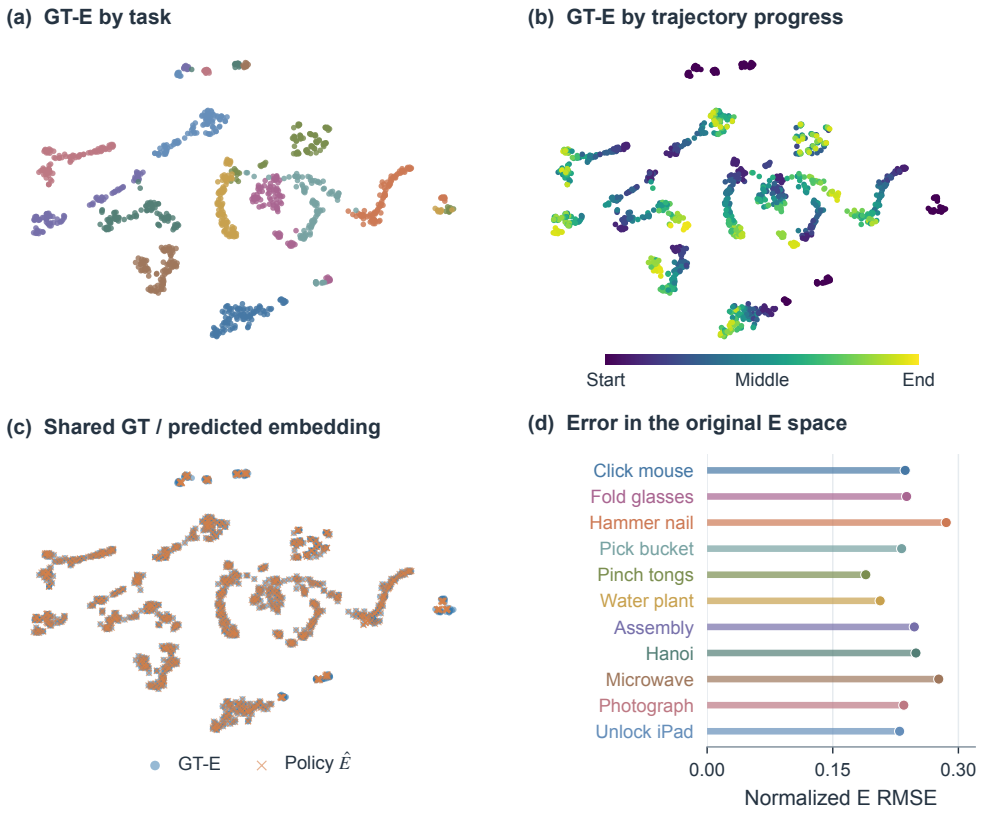



Figure C.1: Structure and prediction error of the effect space. (a) GT-E by task; (b) the same GT-E windows by relative trajectory progress; (c) GT-E and predicted effects in a shared embedding; (d) paired normalized effect RMSE in the original space. PCA uses 50 components (87.27% explained variance); the shared t-SNE uses perplexity 30 and seed 42.

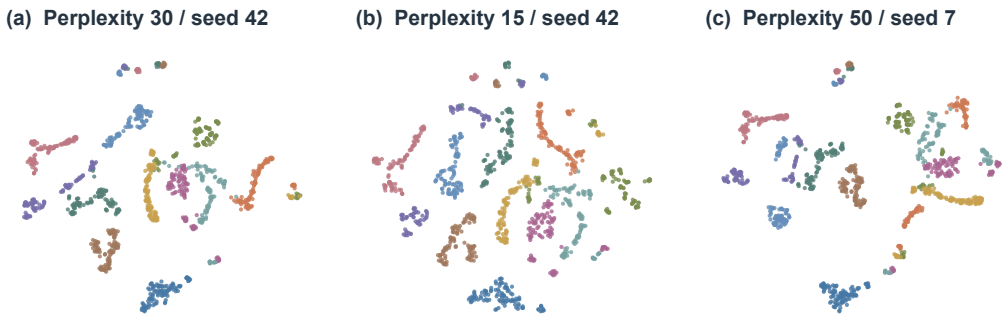



Figure C.2: Sensitivity of the two-dimensional projection. GT-E windows after the same normalization and PCA are projected with t-SNE at perplexity/seed 30/42, 15/42, and 50/7. Task colors match Figure C.1.

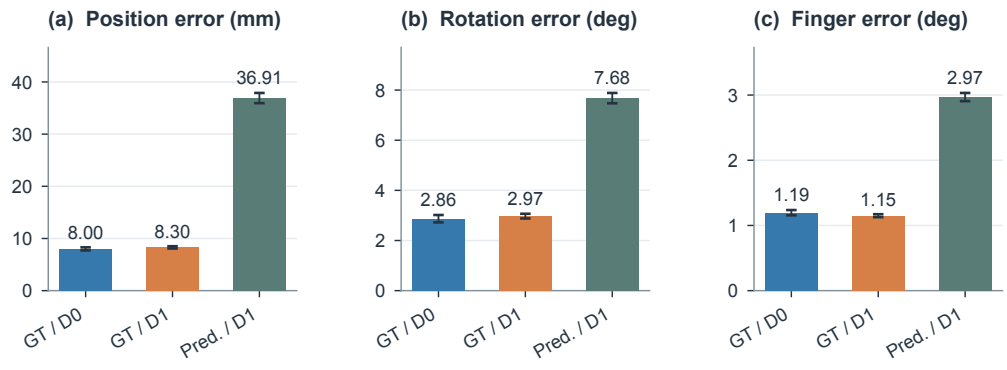



Figure C.3: Physical action-readout error on matched demonstration windows. GT-E/$D_0$, GT-E/$D_1$, and predicted-E/$D_1$ are compared on mean Euclidean position error per hand, rotation geodesic error, and mean absolute finger joint-angle error. Means aggregate ten windows per trajectory over 110 trajectories; intervals are 95% within-task episode-bootstrap intervals (3,000 resamples). $D_0$ is used only for this diagnostic; the deployed policy uses $D_1$.

and in Photograph, progressing from camera grasp to shutter actuation. In the corresponding baseline rollouts, the microwave interaction remains incomplete and the shutter is not triggered. Together, these examples illustrate AFCE's ability to sustain task progress through contact, object transfer, and final actuation, complementing the task-level success rates in Table 1.

# D Reproducibility Details

## D.1 Implementation and Hyperparameters

Table D.1 lists the evaluated AFCE configuration. Stage I uses three 64-dimensional evidence branches and produces a $30 \times 256$ Effect. Its QueryFusion and temporal modules each have two layers, eight attention heads, a feed-forward width of 1024, and zero dropout. The state-conditioned action decoder has four layers with the same width, head count, and feed-forward width.

The Stage-I visual descriptors are projected with PCA fitted on tokenizer training episodes only. The `legacy-none` mask does not exclude robot regions. For Stage II, the original action input and output projections are reinitialized for the Effect width. Non-LoRA language-model weights are frozen. The Stage-I decoder is transferred to the policy implementation with separate single-arm and bimanual output interfaces; the corresponding decoder is trainable from update zero and is also used for execution.

## D.2 Training and Inference Procedure

The two stages follow this procedure:

1. **Train the Effect tokenizer.** Construct synchronized action, response, and sparse visual evidence for each demonstration window. Optimize the Stage-I objective in Eq. equation A.4. Use the tokenizer checkpoint at update 30,000 and report physical reconstruction on the 110 diagnostic trajectories drawn from its training demonstrations. These diagnostics are not held-out validation.
2. **Cache fixed Effect targets.** Encode the demonstration windows using the selected tokenizer and store the resulting Effects in float16. Fit the channel-wise Effect normalization statistics using tokenizer training episodes only. Freeze the tokenizer, cached targets, and Effect, action, and state normalization statistics for Stage II.
3. **Train the Effect policy and decoder.** Sample demonstration windows in proportion to their frame counts. For each batch, sample flow time and Gaussian noise, construct the

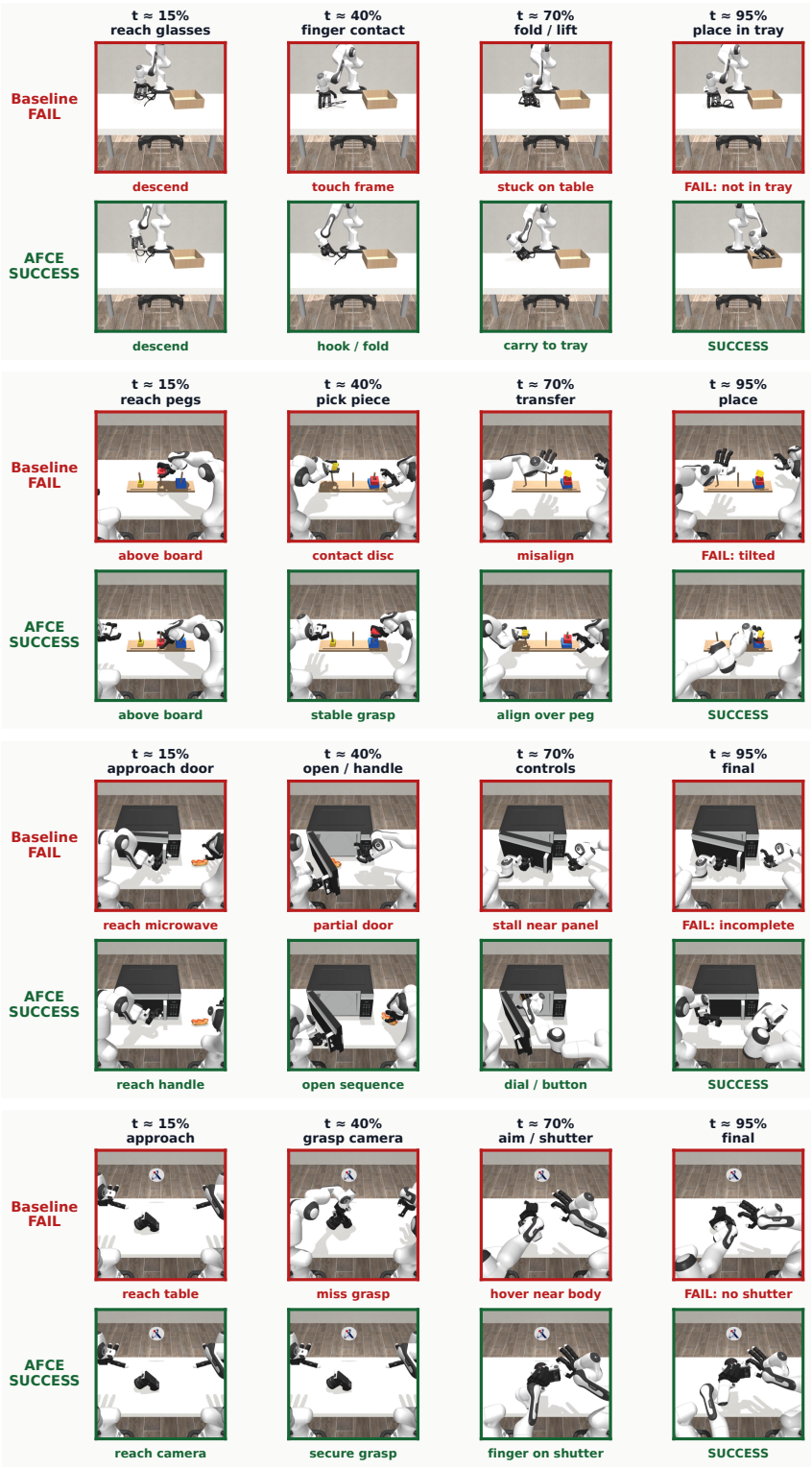



Figure C.4: Illustrative AFCE and $\pi_{0.5}$ rollouts on Fold Glasses, Hanoi, Microwave, and Photograph (top to bottom). Each task is shown at approximately 15%, 40%, 70%, and 95% of the rollout. The selected AFCE rollouts complete their task-specific interaction, whereas the corresponding $\pi_{0.5}$ rollouts fail at contact, placement, or final actuation.

Table D.1: Hyperparameters of the evaluated AFCE configuration.

| **Hyperparameter** | **Value** |
|---|---|
| *Stage I: Effect tokenizer* | |
| Horizon / feature widths | $H = 30$; branch width 64; Effect width 256 |
| Fusion / temporal / decoder layers | $2/2/4$; 8 heads; FFN width 1024; dropout 0 |
| Visual sampling | Stride 5 plus terminal frame; at most 6 dynamic and 2 reference regions, with 8 slots per region |
| Visual descriptor / PCA | 136 dimensions / 64 components; frozen DINOv3; `legacy-none` mask |
| Action reconstruction | Grouped Huber; position weight $w_p = 2$ |
| Auxiliary reconstruction | $\lambda_S(u) = \lambda_V(u) = 0.05 + 0.15 \min\{u/1000, 1\}$ |
| SO(3) loss weight | $\lambda_R = 0.05$ |
| Updates / global batch / seed | $30{,}000/32/42$ |
| AdamW / gradient clip | $(\beta_1, \beta_2) = (0.9, 0.99)$; weight decay 0.01; global norm 1.0 |
| Learning rate | $10^{-4}$ through update 20,000; $3 \times 10^{-5}$ thereafter |
| Effect normalization | Per-channel training-data statistics; standard-deviation floor $10^{-4}$ |
| *Stage II: Effect policy and action decoder* | |
| Base / Effect shape | Official $\pi_{0.5}$ base; normalized Effect $30 \times 256$ |
| Observation / instruction | One image time step; at most 250 language tokens; discrete state input enabled |
| LoRA rank / alpha | PaliGemma 2B: $16/16$; 300M action expert: $32/32$ |
| Flow-time sampling | $\tau = 0.999\xi + 0.001$, $\xi \sim \mathrm{Beta}(1.5, 1)$ |
| GT-action loss | Grouped Huber threshold 1; $\lambda_{\mathrm{GT}} = 0.25628781345139295$ |
| Finger-difference loss | Huber threshold 1; $\lambda_{\Delta f} = 0.05$; per-joint RMS floor 0.1 |
| Auxiliary schedule | $\tau \in [0.2, 0.7]$; linear ramp over 10,000 updates |
| Decoder learning rate | Policy learning rate $\times 0.25$; decoder warm-up 0 updates |
| Updates / global batch / seed | $60{,}000/32/42$ |
| AdamW / gradient clip | $(\beta_1, \beta_2) = (0.9, 0.95)$; $\epsilon = 10^{-8}$; weight decay $10^{-10}$; global norm 1.0 |
| Policy learning rate | Warm-up over 10,000 updates to $5 \times 10^{-5}$; constant thereafter |
| Precision / EMA | Policy bfloat16; decoder and losses float32; no EMA |
| Flow steps at inference | 10 |

noisy Effect, and compute flow matching. Decode the single-step clean Effect estimate and apply the ground-truth-action and finger-difference losses when the flow-time gate is active. Update the policy and the same embodiment decoder jointly using Eq. equation A.9.

4. **Execute the policy.** Starting from noise, perform ten flow steps to predict a normalized Effect. Reverse Effect normalization and pass it, together with the current robot state, to the jointly trained embodiment decoder. Execute the resulting native action chunk with receding-horizon replanning.

For the two Stage-II action losses, the time weight is zero outside $\tau \in [0.2, 0.7]$ and corrects for the probability mass of this interval under the specified flow-time sampling distribution. The GT-action coefficient was fixed after calibration on 16 batches: its value gives an approximately 0.1 weighted median gradient ratio relative to flow matching. Finger-difference scaling uses the batch RMS of demonstrated adjacent differences for each finger joint; gradients do not pass through that scale. Adjacent pairs are included only when both steps satisfy the Effect-valid mask.

Stage-II training uses two nodes with four GH200 GPUs each and a global batch size of 32. Sampling is frame proportional, with a committed data cursor, a PCG64 permutation for each epoch, and `drop_last`. We evaluate the checkpoint obtained after 60,000 training updates. Training uses seed 42. Closed-loop results are evaluated with environment seeds 0, 1, and 2, using 50 episodes per task per seed and one trained checkpoint per method. The reported standard deviations measure variation across evaluation seeds, not across independent training runs. Task definitions, camera selection, episode counts, and evaluation metrics are specified in Sec. B.