LEAPBOT-JAC: JOINT ACTION CHAINS FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Action-chunking policies predict future controls but execute only a prefix before replanning. How should a fixed action-token budget be organized to support execution? We propose LeapBot-JAC, which allocates sparse future-control references and dense execution targets within one shared action expert. Both chains are supervised by demonstrated controls and jointly denoised at aligned physical-time positions. Directed attention allows execution tokens to read reference tokens subject to the host's visibility constraints. JAC adds neither action tokens nor parameters and retains the host's world-modeling process; both chains participate in inference, while only the execution chain drives the robot. On LIBERO, JAC reaches 98.3% success with FastWAM and LeapBot-WA, up from 97.0% and 97.3%. On zero-shot LIBERO-Plus, their success rates improve from 53.5% to 70.4% and from 73.1% to 77.3%, respectively. RoboTwin and two real-world tasks provide additional deployment evaluations. Single-training-seed comparisons and FastWAM ablations support action-role organization as a practical design choice under a fixed action-token budget.

1 INTRODUCTION

Action-chunking policies predict a sequence of future robot controls, but closed-loop deployment often executes only a short prefix before querying the policy again. ACT and Diffusion Policy exploit sequence generation for temporally coherent behavior while using temporal ensembling or receding-horizon execution to incorporate new observations (Zhao et al., 2023; Chi et al., 2023). Predictions beyond the executed prefix can still influence the current output through the model's internal dependencies. The resulting design question is how to organize those predictions: should every token have the same temporal density and deployment role?

World action models (WAMs) couple robot control with learning environment dynamics. Motus supports multiple video–action modeling modes (Bi et al., 2025); LingBot-VA uses causal video–action generation (Li et al., 2026a); Fast-WAM retains video co-training while omitting future-video

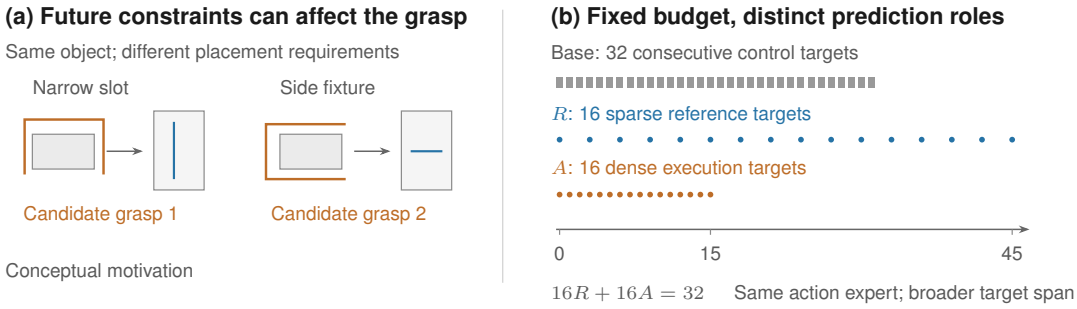

Figure 1: **Motivation and budget allocation.** (a) A conceptual illustration of how future placement constraints may affect current grasp choices; it is not an experimental rollout or a validated feasible solution. (b) JAC reallocates 32 action tokens into sparse reference and dense execution targets on the same control-time axis. Only the execution prefix drives the robot.

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.

generation at inference (Yuan et al., 2026). These approaches primarily organize world representations and their interaction with control. Within an existing action expert, the organization of a fixed number of control predictions remains a separate design choice.

We study **action-role organization under a fixed action-token budget**. Dense predictions preserve near-term control resolution, whereas sparsely sampled control targets span a longer interval using fewer tokens. Combining these roles provides a way to expose execution to a temporally extended reference without adding an action expert or expanding its token sequence. The reference consists of robot controls, not semantic subgoals or executable waypoints; its role is to condition the jointly generated execution chain.

We propose LeapBot-JAC (Joint Action Chains, JAC). A sparse reference chain R and a dense execution chain A share input projections, Transformer layers, an output head, and every action forward pass. Both chains are supervised by demonstrated controls and evolve together during denoising, with independent noise samples at a common noise level. Physical-time positions align the two target sequences. A directed role mask permits A to read R while blocking direct access in the reverse direction, subject to the host's legal attention edges. Only the execution prefix is sent to the robot. This construction reorganizes action prediction while preserving the host's world-modeling process.

Prior work already explores joint generation of high-level intent and executable actions (Li et al., 2026b; Hao et al., 2025). JAC's focus is more specific: directly supervised control references and dense execution coexist within one existing action expert at a fixed token budget. We assess task success across two hosts and examine how information-flow direction, reference weight, and reference stride affect performance.

Our contributions are threefold. **(1) Action-role allocation:** we formulate sparse reference and dense execution prediction within a fixed action-token budget, separating the supervision span from the executed prefix. **(2) Shared-expert joint generation:** we combine control-label supervision, physical-time alignment, and directed cross-chain access without adding action parameters, while retaining host visibility constraints. **(3) Empirical evidence on two hosts:** JAC improves LIBERO success for FastWAM and LeapBot-WA to 98.3%, and improves their LIBERO-Plus totals by 16.9 and 4.2 percentage points, respectively. Three FastWAM ablations characterize sensitivity to access direction, reference supervision, and temporal coverage.

2 Related Work

Action sequences and hierarchical control. ACT improves fine manipulation through sequence generation and temporal ensembling (Zhao et al., 2023). Diffusion Policy represents multimodal action distributions with conditional diffusion and receding-horizon control (Chi et al., 2023). Play-LMP learns latent plans from play data for goal-conditioned behavior (Lynch et al., 2019). Dense Policy progressively expands sparse action keyframes into a dense sequence (Su et al., 2025). JAC instead retains simultaneous reference and execution roles at a fixed total token count. Hierarchical methods distinguish longer-term plans from local execution. CHD jointly diffuses high-level subgoals and low-level trajectories, allowing low-level feedback to refine subgoals (Hao et al., 2025). JAC allocates reference and execution roles inside one existing action expert; its directionality is evaluated as an architectural choice.

World modeling and action generation. Motus combines multiple experts and modality schedules for different conditional generation modes (Bi et al., 2025). LingBot-VA jointly models video and control with causal blocks and history updates (Li et al., 2026a). Fast-WAM separates video co-training from test-time future generation (Yuan et al., 2026), while LeapBot-WA explores world anchoring in predictive semantic space (Liu et al., 2026). JAC preserves a host's world objective and sampling process while changing targets and visibility within its action budget. Beyond joint action generators, DINO-WM optimizes actions through predicted visual features (Zhou et al., 2024), and V-JEPA 2-AC uses an action-conditioned latent world model for planning (Assran et al., 2025).

Intermediate representations and action reasoning. UniPi synthesizes text-conditioned video plans and extracts controls from them (Du et al., 2023); CoTDiffusion generates visual subgoals to guide a low-level policy (Ni et al., 2024). These differ from JAC's directly supervised control refer-

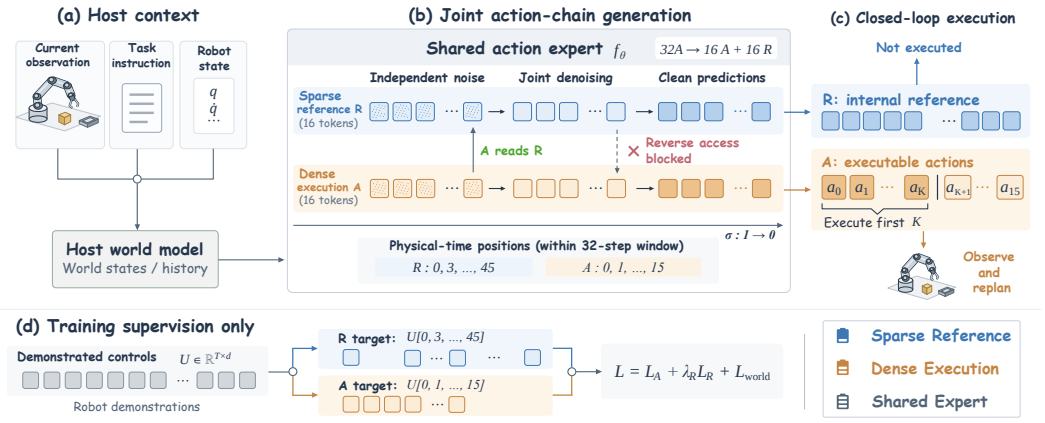

Figure 2: Overview of LeapBot-JAC. A shared action expert jointly denoises a sparse reference chain R and a dense execution chain A within a fixed action-token budget. Physical-time encoding aligns the two chains, while a directed attention mask allows A to attend to R and blocks direct reverse access, subject to the host's visibility constraints. Both chains are supervised by demonstrated controls, and only the execution prefix drives the robot. The illustrated $K = 10$ is used on LIBERO/LIBERO-Plus; RoboTwin uses $K = 16$. The host's world-modeling objectives and sampling schedule are preserved.

ences. ECoT generates embodied reasoning about objects, end-effector states, and subtasks before actions (Zawalski et al., 2024). LAWA learns codebook-based transition targets from action-free video, jointly denoises separate latent-action and executable-action experts, and omits the future video branch at inference (Li et al., 2026b). Its online latent intent evolves in a continuous embedding space. JAC shares the idea that execution can use evolving intent, but differs in **supervision**, taken directly from robot controls without a new video-action tokenizer; **parameters and budget**, reallocating a fixed token count inside an existing expert; and **host integration**, retaining the host world process while changing action-target organization and direct cross-chain access.

Spatial, semantic, and historical conditions. Robust-WAM aligns action-stream queries with future semantic features and their physical times (Yan et al., 2026). MaskWAM combines object-mask prompting and prediction (Yu et al., 2026), and MemoryWAM studies persistent history (Yang et al., 2026). These mechanisms address semantic, spatial, and historical conditioning. JAC complements these mechanisms by supervising intermediate robot controls.

Noise schedules and sequence generation. Diffusion Forcing assigns independent noise levels to sequence tokens (Chen et al., 2024). JAC instead uses independent noise realizations at a common action noise level.

3 Joint Action Chains under a Fixed Budget

3.1 Problem and Fixed-Budget Formulation

At query time t, conditions c_t comprise observations, instructions, proprioception, and host-permitted history. Let $U = [u_0, \ldots, u_{T-1}] \in \mathbb{R}^{T \times d}$ denote demonstrated future controls relative to t. We distinguish supervision-window length T, action-token budget B, and executed prefix K. JAC changes target allocation, physical-time positions, cross-chain visibility, and reference supervision within one action expert.

Let W, C denote host world states and caches, and M_{host} its legal visibility. JAC retains the host world process and requires configurable action targets, physical-time positions, and attention visibility. Reference times must fit within a legally joint generation block; host integration is detailed in Appendix A.8.

3.2 Target Construction and Boundary Padding

A fixed rule π produces ordered index sets I_R, I_A, with $|I_R| = n_R$, $|I_A| = n_A$, and $n_R + n_A = B$. Target allocation is

$$G_\pi(U) = \big(U[I_R], U[I_A], I_R, I_A\big). \tag{1}$$

Execution uses $I_A = \{0, \dots, n_A - 1\}$, with $K \le n_A$, while $I_R = \{0, s, \dots, (n_R - 1)s\}$ sparsely samples future controls. Unless an ablation specifies otherwise, we use $s = 3$, $\lambda_R = 0.2$, $B = 32$, $n_R = n_A = 16$, and $K = 10$ on LIBERO/LIBERO-Plus. RoboTwin uses $K = 16$ for both bases and JAC. The default loader window has $T = 48$ controls; its 16 sampled references end at index 45. The action-token budget remains 32:

$$R_0 = [u_0, u_3, \dots, u_{45}], \quad A_0 = [u_0, u_1, \dots, u_{15}]. \tag{2}$$

Subscript 0 denotes a clean target in normalized action space; indices denote physical control times. Both chains are subsequences of one expert. Across base/JAC and $s = 2, 3, 4$, trajectory starts, video-frame indices and counts, world-prediction steps, and world-loss coefficients remain fixed. Extending reference coverage changes neither world targets nor the executed prefix.

Boundary padding. Out-of-episode indices are clamped to the last valid action and marked by P, with $P_R = P[:, :: 3]$ and $P_A = P[:, : 16]$ at the default stride. Filled targets undergo normal transformations and noising. Padding is excluded from direct supervision but remains attention-visible under the role mask. Appendix A.4 specifies the delta-control fill rule.

Coverage and overlap. For a fully valid window, R and A overlap at $\{0, 3, 6, 9, 12, 15\}$: 32 tokens cover 26 distinct control times. References supervise both overlapping and more distant sampled times; padding reduces valid coverage. In general,

$$O = |I_R \cap I_A|, \qquad D = |I_R \cup I_A| = B - O. \tag{3}$$

Overlap and distinct-target count are coupled at fixed budget. Dense A preserves near-term control resolution; sparse R supplies auxiliary targets over a longer physical span. These incremental controls are predictive references, not executable waypoints or abstract subgoals.

3.3 Shared Generation and Time Alignment

We retain the host's continuous generation parameterization. For linear flow matching (Lipman et al., 2023), each sample uses a common noise level $\sigma \in [0, 1]$ and independent Gaussian samples ϵ_R, ϵ_A:

$$R_\sigma = (1 - \sigma)R_0 + \sigma\epsilon_R, \tag{4}$$

$$A_\sigma = (1 - \sigma)A_0 + \sigma\epsilon_A. \tag{5}$$

The existing action expert receives $X_\sigma = [R_\sigma; A_\sigma]$:

$$[\widehat{v}_R; \widehat{v}_A] = f_\theta(X_\sigma, \sigma, p; W, C, c_t, M). \tag{6}$$

Both chains share input/output projections, time embeddings, Transformer layers, and each action forward pass. Their targets, positions, noise, losses, and visibility distinguish their roles. Independent noise differentiates noisy inputs at overlapping times; shared conditions and parameters still induce dependence.

Figure 3 places both chains on one physical-time axis. Their position vector is

$$p = [0, 3, \dots, 45;\ 0, 1, \dots, 15]. \tag{7}$$

For hosts using RoPE (Su et al., 2021), the relative offset between execution token i and reference token j is $3j - i$, rather than their concatenated-slot offset $j - (16 + i)$. The host frequency table is retained, with no additional chain-identity embedding. Physical durations must be aligned when control frequencies differ.

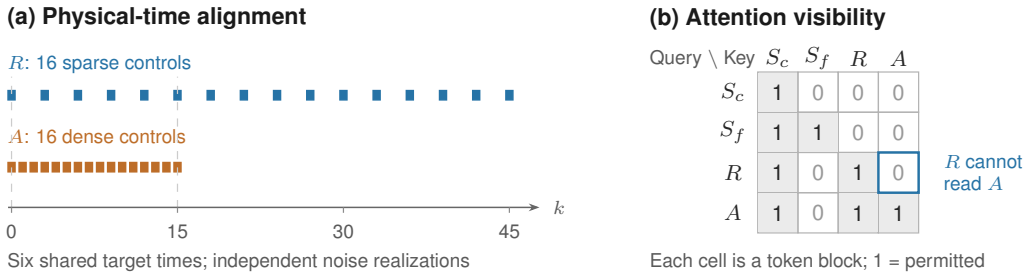

Figure 3: **Time alignment and directed visibility.** (a) At $s = 3$, reference indices span 0–45 and execution indices span 0–15. (b) The outlined 16×16 block prevents R queries from reading A keys. S_c, S_f denote current/future semantics. The full matrix illustrates an action-independent semantic context; non-action restrictions are inherited from the host. General visibility follows Eq. (8).

3.4 Directed Visibility and Training Objective

Mapping host visibility to physical times and legal generation blocks gives

$$M_{\mathrm{JAC}} = M_{\mathrm{host}} \wedge M_{\mathrm{role}}, \quad M_{\mathrm{role}}|_{R,A} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{8}$$

Rows are queries and columns are keys: A reads R, while R cannot read A. For a fully visible 32-token action block, the mask adds one 16×16 zero block. Edges outside the action–action block retain host visibility.

Figure 3 illustrates a host where semantic tokens do not read actions and actions cannot read future semantics. These additional restrictions are inherited from that host.

Local directionality. The mask blocks direct $A \to R$ attention. Indirect reverse paths through host states require the conditions in Appendix A.9 to be excluded; directionality does not imply statistical independence.

Supervision strength. Let $P_{L,k} = 1$ mark padding and $N_L = \sum_k (1 - P_{L,k})$ count valid targets in chain L. The per-sample chain loss averages squared error over action dimensions and valid tokens, with the host's common timestep weight $w(\tau)$:

$$\ell_L = \frac{w(\tau)}{d \max(N_L, 1)} \sum_k (1 - P_{L,k}) \left\| \widehat{v}_{L,k} - (\epsilon_{L,k} - L_{0,k}) \right\|_2^2, \qquad L \in \{R, A\}. \tag{9}$$

Here τ is the sampled training timestep, distinct from environment time t, and σ is its scheduler-mapped noise level. Both chains share τ and $w(\tau)$; a chain with no valid targets contributes zero direct loss. The training objective is

$$\mathcal{L} = \mathbb{E}[\ell_A + \lambda_R \ell_R] + \mathcal{L}_{\mathrm{host,world}}, \tag{10}$$

where the expectation averages over samples, timesteps, and noise; world-loss coefficients are unchanged. With execution weight 1, λ_R controls direct reference supervision, not attention strength. Overlapping times are supervised in both roles. Valid-token averaging removes direct scaling with target count, while padding can still alter coverage and predictions (Appendix A.5).

Execution losses also backpropagate through reference representations, including padded positions. Thus $\lambda_R = 0$ would retain reference computation and indirect supervision, and is not equivalent to the base.

3.5 Inference and Resource Accounting

Each query initializes world states and caches as in the host, and R, A with independent Gaussian noise. Both chains are jointly integrated from $\sigma = 1$ to 0 in normalized action space. Only $\widehat{A}$ is inverse-transformed, and its first K controls are executed before replanning. Inference has no

demonstration-tail padding mask: all reference and execution slots are generated at every query. Every reference slot participates in each action-expert update even when the reference trajectory is not returned.

Algorithm 1 in Appendix A.10 summarizes the sampling procedure.

At fixed B, JAC preserves action parameter count, tokens per step, and action-expert forward counts while retaining host world updates. Structural equality does not by itself imply equal runtime; model-call latency and memory accounting are addressed in Section 4.4.

4 EXPERIMENTS

We evaluate whether JAC improves two WAM hosts, FastWAM and LeapBot-WA, across standard manipulation, zero-shot perturbations, and bimanual tasks. Controlled LIBERO ablations examine visibility, supervision, and reference stride; two real-world tasks test physical deployment.

4.1 EXPERIMENTAL SETUP

Configuration and evaluation. Unless otherwise specified, JAC uses $16R+16A$, reference stride $s = 3$, supervision weight $\lambda_R = 0.2$, and directed $R \to A$ access. Our local Base/JAC comparisons fix the total action-token count at 32. LIBERO/LIBERO-Plus use $K = 10$; RoboTwin bases and JAC are all re-evaluated with $K = 16$, matching JAC's execution-chain length. LIBERO (Liu et al., 2023) reports success on Spatial, Object, Goal, and Long, with their four-suite mean. For LIBERO-Plus (Fei et al., 2026), we directly evaluate LIBERO-trained policies without Plus training, fine-tuning, or test-time parameter adaptation. Its seven perturbation categories measure robustness of learned tasks; our Total aggregates all test episodes. RoboTwin 2.0 (Chen et al., 2025) reports Clean, Random, and their mean.

Baselines and reporting. We compare our reproduced FastWAM and LeapBot-WA bases with their JAC variants. External references are π_0, $\pi_{0.5}$, Motus, LingBot-VA, LAWA, and ImageWAM, using available results per benchmark. Table citations identify the numerical source, prioritizing original reports and using later evaluations when needed. Literature results provide context under different training recipes and model budgets; within-host pairs assess JAC. Parentheses denote absolute percentage-point changes from each reproduced base. We follow each host paper's training hyperparameters (Yuan et al., 2026; Liu et al., 2026), with the JAC and execution settings specified here. Training and evaluation use an eight-NVIDIA-H100 system. Appendix A.3 summarizes the protocol.

4.2 MAIN RESULTS

LIBERO: standard manipulation. Table 1 reports the results on the four LIBERO suites. JAC raises the average success rate from 97.0 to 98.3 on FastWAM and from 97.3 to 98.3 on LeapBot-WA, corresponding to gains of 1.3 and 1.0 percentage points, respectively. Both hosts improve by 1.6 points on Long, while LeapBot-WA shows a slight 0.2-point decrease on Goal. Other gains vary by host: the largest increases occur on Goal for FastWAM (+2.0 points) and Spatial for LeapBot-WA (+2.6 points).

RoboTwin 2.0: bimanual manipulation. Our models use both clean and randomized demonstrations. When executing the first 16 predicted actions from each action chunk, JAC raises Avg. from 91.02 to 91.24 on FastWAM ($+0.22$) and from 91.49 to 92.00 on LeapBot-WA ($+0.51$). FastWAM gains 0.56 points on Random but loses 0.12 on Clean; LeapBot-WA gains 0.54 and 0.48 on Clean and Random.

LIBERO-Plus: zero-shot robustness. Table 2 reports the seven perturbation categories. JAC improves all seven on both hosts, raising Total from 53.5 to 70.4 on FastWAM ($+16.9$) and from 73.1 to 77.3 on LeapBot-WA ($+4.2$). FastWAM's largest gains occur under Noise ($+34.1$) and Camera ($+24.5$), whereas LeapBot-WA improves most under Robot ($+5.7$), followed by Background and Noise ($+5.1$ each). The broader improvements under perturbation complement the smaller clean-LIBERO gains.

Table 1: **Standard manipulation.** Success rates (%) on LIBERO and RoboTwin 2.0. Cited results and reproduced Base/JAC pairs are separated. Parentheses denote the improvement in average success rate (pp); bold indicates the better result within each reproduced pair. Our RoboTwin experiments execute 16 actions from each predicted action chunk; literature protocols are retained as reported. Motus LIBERO scores are reported by Yuan et al. (2026); its RoboTwin scores follow Bi et al. (2025).

	LIBERO					RoboTwin 2.0		
Model	**Spa.**	**Obj.**	**Goal**	**Long**	**Avg.**	**Clean**	**Random**	**Avg.**
Literature reports								
π_0 (Yuan et al., 2026)	96.8	98.8	95.8	85.2	94.1	65.92	58.40	62.16
$\pi_{0.5}$ (Yuan et al., 2026)	98.8	98.2	98.0	92.4	96.9	82.74	76.76	79.75
Motus (Bi et al., 2025)	96.8	99.8	96.6	97.6	97.7	88.66	87.02	87.84
LingBot-VA (Li et al., 2026a)	98.5	99.6	97.2	98.5	98.5	92.93	91.55	92.24
ImageWAM (Zhang et al., 2026)	97.2	99.2	98.8	98.4	98.4	93.20	93.56	93.38
Our reproductions								
FastWAM	96.8	99.4	96.2	95.6	97.0	**91.44**	90.60	91.02
+ JAC	**98.0**	**99.6**	**98.2**	**97.2**	**98.3** (+1.3)	91.32	**91.16**	**91.24** (+0.22)
LeapBot-WA	96.2	99.6	**97.6**	95.6	97.3	90.92	92.06	91.49
+ JAC	**98.8**	**99.8**	97.4	**97.2**	**98.3** (+1.0)	**91.46**	**92.54**	**92.00** (+0.51)

Table 2: **Zero-shot LIBERO-Plus.** Success rates (%) without Plus adaptation. Aggregate is full-test Total for our rows and trial-level aggregation for π_0/LAWA; † retains ImageWAM's reported Avg.; equivalence to our episode-weighted Total is not assumed. Parentheses denote changes in aggregate success rate (pp); bold indicates gains.

Model	**Cam.**	**Robot**	**Lang.**	**Light**	**Bg.**	**Noise**	**Layout**	**Agg.**
Literature reports								
π_0 (Li et al., 2026b)	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
LAWA (Li et al., 2026b)	69.2	66.7	62.8	96.2	64.5	85.5	78.6	74.4
ImageWAM (FLUX.2 4B) (Zhang et al., 2026)	80.8	50.3	91.4	98.1	85.5	93.8	80.5	83.1†
Our reproductions								
FastWAM	19.2	46.2	69.9	82.0	53.7	41.0	62.8	53.5
+ JAC	**43.7**	**59.9**	**82.5**	**92.7**	**62.7**	**75.1**	**76.3**	**70.4** (+16.9)
LeapBot-WA	33.8	75.1	87.8	93.4	90.1	54.7	76.7	73.1
+ JAC	**37.3**	**80.8**	**91.7**	**95.9**	**95.2**	**59.8**	**80.3**	**77.3** (+4.2)

4.3 Ablation Studies on LIBERO

All three groups use FastWAM with a fixed $16R{+}16A$ JAC budget. The visibility study is conducted with $s = 3$ and $\lambda_R = 0.2$; the stride sweep fixes $\lambda_R = 0.2$; and the measured weight sweep uses $s = 2$, rather than the default stride-3 setting. Each group therefore examines one design factor while keeping the remaining settings fixed.

G1: cross-chain visibility. Directed $R \to A$ access achieves the best performance at 98.3, compared with 98.2 for reverse access, 97.7 with no direct cross-chain access, and 97.5 for bidirectional access (Table 3). Among the tested visibility patterns, the directed $R \to A$ design provides the strongest overall result, supporting an asymmetric information flow in which the execution chain can leverage the reference chain while the reference chain remains independent of the execution chain. We therefore adopt this directed reference-to-execution visibility as the default JAC design.

G2: reference-supervision weight. We vary the reference-supervision weight at $s = 2$. Setting λ_R to $0.1, 0.2, 0.5$, and 1.0 yields success rates of 97.8, 98.1, 97.5, and 96.2, respectively (Figure 4a).

Table 3: G1: visibility on LIBERO at $s = 3$, $\lambda_R = 0.2$. Permissions refer to direct attention before host intersection. Padded targets are loss-masked but remain attention-visible.

Information flow	A **reads** R	R **reads** A	**Spatial**	**Object**	**Goal**	**Long**	**Avg.**
$R \leftrightarrow A$	$\surd$	$\surd$	98.0	99.4	97.0	95.4	97.5
$R \rightarrow A$	$\surd$	$\times$	98.0	99.6	98.2	97.2	**98.3**
No direct cross-chain	$\times$	$\times$	98.6	100.0	98.0	94.2	97.7
$A \rightarrow R$	$\times$	$\surd$	98.4	99.6	98.0	96.6	98.2

Performance peaks at the moderate setting $\lambda_R = 0.2$, while larger weights lead to progressively lower results, suggesting that the reference objective is most effective when used as a complementary supervision signal rather than dominating the action objective. The 97.0 FastWAM baseline is shown separately for reference and is not part of the weight sweep.

G3: reference stride. With $\lambda_R = 0.2$, strides 2, 3, and 4 achieve 98.1, 98.3, and 98.0, respectively (Figure 4b), all outperforming the 97.0 FastWAM baseline. The best result is obtained at $s = 3$, corresponding to a 1.3-point gain over the base model. The consistently strong performance across all three strides further shows that JAC is robust to the exact temporal spacing of the reference chain, while the intermediate stride provides the most favorable balance between temporal coverage and reference density.

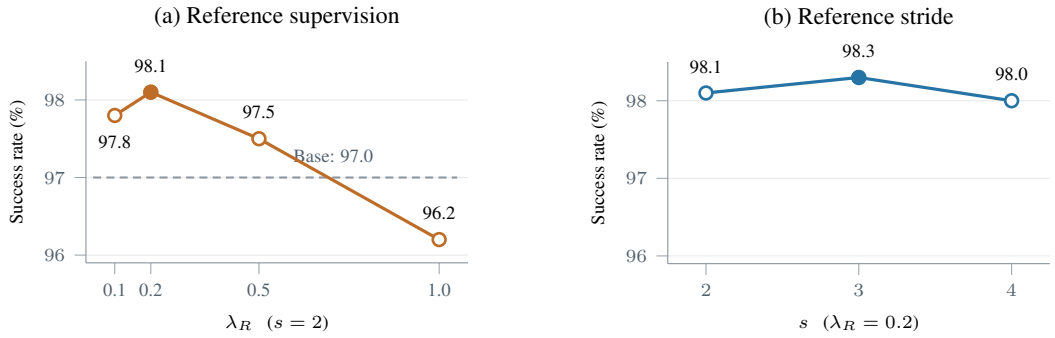

Figure 4: **LIBERO sensitivity on FastWAM.** (a) Reference-supervision weight at $s = 2$, with the no-JAC baseline shown as a separate dashed reference. (b) Reference stride at $\lambda_R = 0.2$; the observed range is 0.3 pp. Measured settings share a restricted vertical scale; lines aid readability. Results use one training seed, without error bars.

4.4 Computational Cost

JAC reallocates the original $32A$ sequence into $16R + 16A$ within the shared action expert, without adding trainable parameters or denoising steps. As shown in Table 4, this restructuring introduces negligible inference overhead on both FastWAM and LeapBot-WA, with essentially unchanged latency and peak memory usage. These results indicate that JAC improves action organization while preserving the computational footprint of the original host models.

4.5 Real-World Evaluation

We evaluate two physical manipulation tasks (Figure 5). **T1: sequential sorting** requires placing an apple into the green basket, followed by placing a green block into the blue basket. Success requires both placements in the specified order. **T2: drawer closing** requires fully closing a drawer whose cabinet can shift during pushing, introducing motion of the manipulated fixture.

Each model is evaluated for four rounds of 50 trials per task (200 trials per model per task), using one training seed; success rates are averaged across rounds. On sequential sorting / drawer closing, FastWAM achieves 82% / 77%, increasing to 93% / 79% with JAC; LeapBot-WA achieves 90% / 85%,

Table 4: **Computational cost within each host.** Model-call latency is wall-clock p50, including host-side scheduling and CPU–GPU transfers within inference. Measurements use the training hardware and configuration. Memory is peak allocated GPU memory.

Backbone	Configuration	Total params (M)	Denoising steps	Latency (ms)	Peak memory (GB)
FastWAM	Base ($32A$)	11701.62	10	337.0	25.00
	JAC ($16R + 16A$)	11701.62	10	337.7	25.00
LeapBot-WA	Base ($32A$)	8428.11	10	438.6	26.13
	JAC ($16R + 16A$)	8428.11	10	435.1	26.13

increasing to 95% / 90%. The corresponding two-task averages, rounded to integer percentages, are 80% / 86% for FastWAM / FastWAM + JAC and 88% / 93% for LeapBot-WA / LeapBot-WA + JAC. These results show gains on both tasks and hosts.

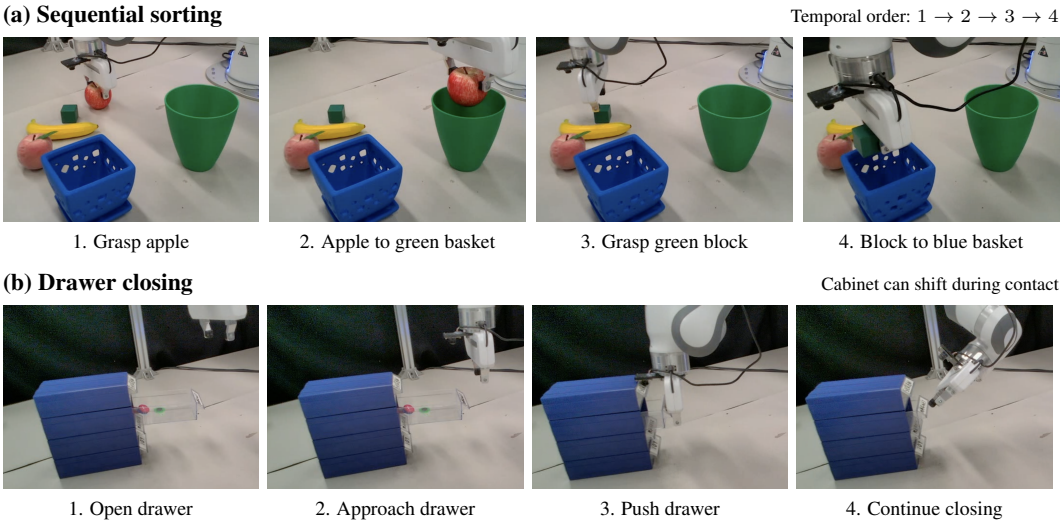

Figure 5: **Real-world task sequences,** ordered left to right. (a) Apple-to-green-basket placement precedes green-block-to-blue-basket placement. (b) Drawer closing with possible cabinet displacement during contact.

The repeated 98.3 main/visibility/stride score denotes one shared configuration, not independent replication. LIBERO-only ablations do not isolate the source of LIBERO-Plus gains.

5 Discussion and Limitations

Reference stride and λ_R jointly affect multiple training factors, limiting causal isolation in the current ablations. In addition, the sparse reference controls are not explicitly constrained to encode semantic intent, leaving their interpretation and broader transfer for future study.

6 Conclusion

We present LeapBot-JAC, which reorganizes a fixed action-token budget into sparse reference and dense execution controls within a shared action expert. Across FastWAM and LeapBot-WA, LeapBot-JAC improves performance on LIBERO, LIBERO-Plus, RoboTwin 2.0, and real-world tasks, with the largest gains under zero-shot perturbations. Ablations validate the main design choices, while matched inference shows negligible overhead. Overall, LeapBot-JAC strengthens world-action models without additional model capacity or inference stages.

AI USE STATEMENT

In preparing this manuscript, generative AI tools assisted with drafting and English translation, literature discovery, clarification of method descriptions, feedback on experimental controls, figure editing, and LaTeX formatting. No empirical results were generated to replace experiments. The authors take responsibility for the final manuscript, including all AI-assisted text, figures, and claims.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

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. *arXiv preprint arXiv:2512.13030*, 2025. URL https://arxiv.org/abs/2512.13030.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2407.01392.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

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. URL https://arxiv.org/abs/2303.04137.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning Universal Policies via Text-Guided Video Generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2302.00111.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/papers/Fei_LIBERO-Plus_A_Progressive_Robustness_Benchmark_for_Visual-Language-Action_Models_CVPR_2026_paper.pdf.

Ce Hao, Anxing Xiao, Zhiwei Xue, and Harold Soh. CHD: Coupled Hierarchical Diffusion for Long-Horizon Tasks. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 1420–1451, 2025. URL https://proceedings.mlr.press/v305/hao25a.html.

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

Xiang Li, Yupeng Zheng, Songen Gu, Huailiang Ma, Feng Yu, Yuhang Zheng, Xian Nie, Shanshuai Yuan, Yujie Zang, Weize Li, Shuai Tian, Moyang Liu, Ya-Qin Zhang, and Wenchao Ding. Latent action as intention enables efficient future imagination for world action models. *arXiv preprint arXiv:2608.24882*, 2026b. URL https://arxiv.org/abs/2608.24882v2.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.03310.

Pei Liu, Nan Zheng, Lang Zhang, Daojie Peng, Yanan Zhang, Feilong Kong, Mingyue Feng, Jiachao Liu, Yaonong Wang, Qifeng Chen, and Jun Ma. LeapBot-WA: World-anchor action models via predictive latent alignments. *arXiv preprint arXiv:2607.23969*, 2026. URL https://arxiv.org/abs/2607.23969.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. *arXiv preprint arXiv:1903.01973*, 2019. URL https://arxiv.org/abs/1903.01973.

Fei Ni, Jianye Hao, Shiguang Wu, Longxin Kou, Jiashun Liu, Yan Zheng, Bin Wang, and Yuzheng Zhuang. Generate Subgoal Images before Act: Unlocking the Chain-of-Thought Reasoning in Diffusion Model for Robot Manipulation with Multimodal Prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13991–14000, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Ni_Generate_Subgoal_Images_before_Act_Unlocking_the_Chain-of-Thought_Reasoning_in_CVPR_2024_paper.html.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. URL https://arxiv.org/abs/2104.09864.

Yue Su, Xinyu Zhan, Hongjie Fang, Han Xue, Hao-Shu Fang, Yong-Lu Li, Cewu Lu, and Lixin Yang. Dense Policy: Bidirectional Autoregressive Learning of Actions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14486–14495, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Su_Dense_Policy_Bidirectional_Autoregressive_Learning_of_Actions_ICCV_2025_paper.html.

Haodong Yan, Junfeng Li, Junjie He, Zhide Zhong, MingMing Yu, Wenxuan Song, Jiaguan Zhu, Yangyang Zheng, Yuqiao Du, Jiadi You, Yingjie Cai, Xu Yan, Guanyi Zhao, Bingbing Liu, and Haoang Li. Robust-WAM: Bridging generative pretraining and semantic foresight in world-action models. *arXiv preprint arXiv:2608.05903*, 2026. URL https://arxiv.org/abs/2608.05903.

Sizhe Yang, Juncheng Mu, Tianming Wei, Chenhao Lu, Xiaofan Li, Linning Xu, Zhengrong Xue, Zhecheng Yuan, Dahua Lin, Jiangmiao Pang, and Huazhe Xu. MemoryWAM: Efficient world action modeling with persistent memory. *arXiv preprint arXiv:2606.20562*, 2026. URL https://arxiv.org/abs/2606.20562.

Hanyang Yu, Haitao Lin, Jingbo Zhang, Wenyao Zhang, Chenghao Gu, Heng Li, and Ping Tan. MaskWAM: Unifying mask prompting and prediction for world-action models. *arXiv preprint arXiv:2606.13515*, 2026. URL https://arxiv.org/abs/2606.13515.

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

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024. URL https://arxiv.org/abs/2407.08693.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. ImageWAM: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026. URL https://arxiv.org/abs/2606.19531.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. URL https://arxiv.org/abs/2304.13705.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. *arXiv preprint arXiv:2411.04983*, 2024. URL https://arxiv.org/abs/2411.04983.

A Experimental Details and Reporting Scope

A.1 Configurations and Target Spans

All reported JAC main configurations use $n_R = n_A = 16$, $s = 3$, and $\lambda_R = 0.2$. The reference indices are $\{0, 3, \ldots, 45\}$ and execution indices are $\{0, \ldots, 15\}$. For a fully available target window, six times overlap, giving 26 distinct controls across 32 tokens. The comparison protocol executes $K = 10$ controls on LIBERO/LIBERO-Plus and $K = 16$ on RoboTwin, with the same replanning interval for each base/JAC pair. The method and Figure 3 use this same default. The measured weight ablation at $s = 2$ is an explicitly identified exception.

Table 5: Reported JAC configuration groups. All use FastWAM unless another host is listed. All results use one training seed.

Evaluation	Host	Stride	Reference weight
Main performance	FastWAM; LeapBot-WA	3	0.2
G1 visibility	FastWAM	3	0.2
G2 weight	FastWAM	2	0.1, 0.2, 0.5, 1
G3 stride	FastWAM	2, 3, 4	0.2

Each host is reproduced using the training hyperparameters in its original paper (Yuan et al., 2026; Liu et al., 2026), with the JAC target, position, visibility, and loss changes specified in Section 3. All local results use one common training seed. Training and evaluation run on a system with eight NVIDIA H100 GPUs. RoboTwin uses both clean and randomized demonstrations; the four local models are re-evaluated with replan$= 16$. Literature rows retain their published protocols and are not treated as matched-budget experiments.

A.2 Reference-Stride Eligibility

For $I_R = \{0, s, \ldots, 15s\}$, strides 2, 3, and 4 require minimum future-label spans of 31, 46, and 61 controls. The confirmed loader window for the default $s = 3$ is 48. With fully available labels, their overlaps with the execution interval are 8, 6, and 4, giving 24, 26, and 28 distinct target times. These are target-layout counts, not guaranteed counts of real labels at trajectory boundaries.

All stride conditions use identical trajectory starts. When a requested future control is unavailable, the sequence is padded; no common 61-valid-control filter is applied to select a different cohort for this comparison. Thus the starting-state distribution is held fixed, while reference span, target overlap, and the frequency of boundary padding can change together. The confirmed fill and masking rules are described in Appendix A.4. Padding frequency was not quantified by stride. The non-monotonic stride results therefore should not be attributed solely to coverage. They also do not establish superiority over dense repeated near-term targets.

A.3 Experimental Settings

Host-specific optimization, training schedules, and baseline hyperparameters follow FastWAM and LeapBot-WA (Yuan et al., 2026; Liu et al., 2026). Table 6 records the shared infrastructure and JAC-specific settings. Single-seed results do not characterize training-seed variability.

A.4 Padding Implementation

For each requested future index, the loader clamps an out-of-range index to the final legal action in that demonstration and records the position in `action_is_pad`. For LIBERO, `delta_action_dim_mask` sets the first six components of padded controls to zero; the non-delta gripper component retains its final legal value. The loader then applies the normal action transformations, dataset normalization, and action-dimension merging to both valid and padded controls. Padding thus has an ordinary target tensor in normalized action space, and its delta components need not be numerically zero there.

Table 6: Implementation and evaluation settings. Host-specific hyperparameters follow the cited original papers; execution settings below apply to our paired comparisons.

Parameter	Setting
Host training hyperparameters	Original FastWAM / LeapBot-WA recipes
Training and evaluation hardware	Eight NVIDIA H100 GPUs
Training seeds	One common seed across local experiments
Action-token budget	Base: $32A$; JAC: $16R + 16A$
Default reference settings	$s = 3$, $\lambda_R = 0.2$, $R \to A$
Weight sweep	$s = 2$; $\lambda_R \in \{0.1, 0.2, 0.5, 1\}$
LIBERO / LIBERO-Plus replan	10 executed actions
RoboTwin replan	16 executed actions for all local models
RoboTwin training data	Clean and randomized demonstrations
LIBERO-Plus evaluation	Zero-shot; full-test episode aggregation
Default loaded action window	48 controls; reference indices $0, 3, \ldots, 45$
World branch across $s = 2, 3, 4$	Fixed baseline frame indices, counts, steps, loss weights
Real-world evaluation	4 rounds $\times$ 50 trials per model per task

The default full window contains 48 controls. Reference targets and masks select indices $0, 3, \ldots, 45$; execution targets and masks select $0, 1, \ldots, 15$. Padding slots undergo the same flow noising and network forward pass, but the factors $1 - P_{L,k}$ in Eq. (9) remove their direct target errors from the loss. Each chain loss averages over action dimensions and its valid tokens, using a denominator clamped to at least one token before applying the shared timestep weight. If a chain has no valid targets, its direct loss is zero.

The padding mask is not incorporated into the Action DiT self-attention key-padding mask. Padded references can serve as keys and values to permitted reference and execution queries. They can consequently influence valid predictions and receive indirect gradients through those predictions. This fixed-length treatment must be distinguished from deleting padded tokens or excluding them from attention.

A.5 Loss Accounting

For each sample and chain, the implementation first averages elementwise squared flow error over the d action dimensions, then sums over non-padding tokens and divides by $\max(N_L, 1)$. It multiplies the result by the common timestep weight $w(\tau)$ and finally averages over the batch. This reduction yields Eq. (9); it is not an unnormalized masked sum.

The inherited scheduler uses a Gaussian-shaped weight centered at the midpoint of its training timestep range:

$$w(\tau) = \frac{\exp\left[-2\left((\tau - N/2)/N\right)^2\right] - y_{\min}}{c}, \qquad N = 1000, \tag{11}$$

where $y_{\min}$ and c are the scheduler's offset and normalization constants, giving approximately unit mean under its training sampling distribution. Both chains use the same sampled timestep and scheduler-mapped noise level. This weighting is inherited from the host, rather than introduced by JAC; its approximately unit mean does not imply an unchanged expected weighted loss, since residuals depend on timestep.

With execution coefficient 1 and fixed world-loss coefficients, λ_R weights the normalized reference objective relative to the execution objective. Valid-token averaging removes the direct multiplicative effect of valid-token count, but does not guarantee equal loss magnitudes or gradient norms across chains. Padding can alter which physical times receive supervision and can affect predictions through attention-visible tokens. Changing λ_R also changes the action objective relative to the world objective; separate role-weighted means are not generally equivalent to a uniformly weighted single-chain mean.

No loss-reweighted single-chain control or zero-weight JAC result is reported. Consequently the base/JAC gain reflects the combined effects of target allocation, visibility, positions, and loss weight-

ing. The weight sweep supports sensitivity to the tested positive coefficients; it does not establish the necessity of direct reference supervision. The no-JAC base remains a separate comparator rather than an assumed zero-weight endpoint.

A.6 Metrics and Run Identity

LIBERO Avg. is the equally weighted mean of Spatial, Object, Goal, and Long, rounded to one decimal place. LIBERO-Plus Total aggregates the full test evaluation and is not reconstructed by averaging the seven displayed categories. All local results use a single common training seed. LIBERO-Plus category counts need not be equal, so its Total is not a macro-average of the displayed categories. Reported differences are descriptive, not significance estimates. Real-world rates average four equal-size evaluation rounds of 50 trials per model per task; these rounds do not constitute independent training seeds.

The main FastWAM + JAC aggregate at $s = 3$, $\lambda_R = 0.2$ is 98.3/70.4, consistent with the corresponding visibility and stride rows. These appearances do not constitute independent replications. The $s = 2$, $\lambda_R = 0.2$ weight/stride result remains 98.1/63.0. No conclusions about seed variability follow from the displayed decimal precision. Neither inference latency nor peak-memory improvements are claimed from fixed token counts.

A.7 RoboTwin Evaluation Protocol

All four local models use clean and randomized training demonstrations and are evaluated with replan$=16$. FastWAM achieves 91.44/90.60/91.02 on Clean/Random/Avg., versus 91.32/91.16/91.24 with JAC. LeapBot-WA achieves 90.92/92.06/91.49, versus 91.46/92.54/92.00 with JAC. Table 1 reports these completed paired comparisons.

A.8 Host Integration

JAC constructs reference/execution targets, concatenates their noisy states, maps physical-time positions, applies a role mask intersected with host legality, and separates the resulting outputs within the existing action expert. The local current-semantic-cache implementation freezes the V-JEPA encoder and its adapter while training the action expert, proprioception encoder, and JEPA context projector used for semantic anchoring. The host-specific training configuration follows its original recipe; this semantic-cache description is not assumed to specify every host.

In both evaluated hosts, video-frame indices, frame counts, future world-prediction steps, and world-loss coefficients are identical between base and JAC and remain fixed across reference strides $s = 2, 3, 4$. The data pipeline retains each host's baseline video-sampling grid when the future-action window is extended. Thus a longer reference span does not introduce additional world targets.

Figure 3 depicts an action-independent semantic context. In hosts with feedback through world tokens or caches, direct role masking does not guarantee global independence. Table 4 measures model-call latency, including host-side scheduling and CPU–GPU transfers within inference, rather than a complete robot observation-to-command cycle. Action-token counts alone do not determine runtime.

A.9 Forward-Dependency Analysis

Suppose that all context visible to R within a query is independent of the execution-chain initialization, and no token mixing, cache update, or solver operation introduces a reverse path. At fixed parameters and conditions, induction over layers and solver updates gives

$$\frac{\partial R^{(q)}}{\partial A^{\text{init}}} = 0, \tag{12}$$

where q indexes solver updates. A fixed, action-independent semantic cache satisfies these conditions. This is a conditional statement about forward computation, not statistical independence: the chains share conditions and parameters, and execution losses can backpropagate through reference representations. It also does not exclude effects of past executed actions on later observations. In

coupled hosts, an $A \to W \to R$ path may violate the stated condition; removing that feedback or copying context would change the host and require separate cost reporting.

A.10 Sampling Procedure

The two chains are updated jointly within the existing action expert; only the execution prefix is sent to the environment. The host schedule determines any world-state and cache updates.

Algorithm 1 JAC sampling with the host process retained

Require: Conditions c_t, host schedule, indices I_R, I_A
1: Initialize world states W and history C as in host
2: Independently sample $R, A \sim \mathcal{N}(0, I)$
3: **for** each update specified by the host **do**
4: Map physical-time positions; compute Eq. (8)
5: Replace action inputs with $[R; A]$ in host modules
6: Jointly predict and update R, A on action updates
7: Update world states and caches on the host schedule
8: **end for**
9: Inverse-transform A; execute its first K controls

B Limits of Mechanistic Interpretation

The three ablations assess direct visibility, positive reference-loss weights, and temporal strides. They support configuration sensitivity, but do not identify reference semantics or prove abstract planning. Sparse controls may act as auxiliary predictive targets, and blocked direct access may retain indirect communication. Isolating target duplication, reference content, supervision allocation, and optimization effects requires additional matched controls; no unperformed mechanism study is claimed as evidence.