HETEROGENEOUS ACTION CACHING: FREEING THE ACTION SOLVER OF WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) are markedly more robust under distribution shift than present-only policies, yet they are costly to run. The common practice of storing the future representation once lets a single video pass serve every action step, but it leaves the dominant cost untouched: once the future context exists, the *action solver* absorbs 84% of the runtime (165.27 of 195.29 ms on FASTER-WAM), against 25.60 ms for the visual branch and 6.04 ms for VAE encoding. The solver has been neglected as a caching target at the granularity that matters: latent-space and block-level caches act on spatially resolved features, while action-policy caches reuse whole chunks or demand an offline schedule or a learned gate. We present HAC (**H**eterogeneous **A**ction **C**aching), which carries curvature-style heterogeneous token caching into action space. It treats each of the 32 chunk positions as a token, derives a scale-free curvature of its denoising path from the three most recent full evaluations, and forecasts each skipped step with a token-specific rule—copy for stable tokens, first-order extrapolation for linear tokens, and a smoothstep-damped blend for chaotic tokens—while a chaotic-only drift budget decides when a genuine evaluation is required. HAC trains nothing, searches nothing offline, and adds no learned parameters. It removes six of ten solver evaluations, speeding up the pipeline by $1.63\times$ ($1.86\times$ on the solver) over FASTER-WAM at $\eta{=}12$: $+0.09$ points on RoboTwin 2.0 and $+0.35$ on LIBERO (seed 42), and -0.22 points on the matched LIBERO-Plus OOD set. The drift budget η acts as a single speed–quality knob, lowering latency monotonically from 8 to 12 while leaving in-distribution success within noise.

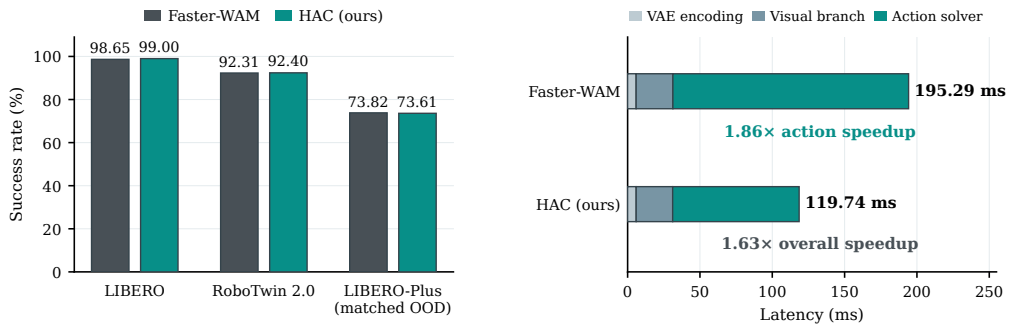
1 INTRODUCTION

World Action Models (WAMs) couple a generative video backbone with an action expert to anticipate how a scene will evolve, and they retain their advantage under camera, object, language, and lighting shift (Zhao et al., 2026a; Yuan et al., 2026; Li et al., 2026b; Bi et al., 2025; NVIDIA, 2025). That robustness is paid for at every control step: a WAM reasons about future dynamics and then repeatedly refines an action chunk, and this solver computation sits on the critical path, limiting whether a robust policy can run at control frequency. The practical question is therefore not whether a WAM should look ahead, but how much of its already-computed reasoning it must recompute before each action.

> *Once the future context is frozen, the action solver should not need to recompute every chunk position at every denoising step.*

The standard response has been to cache the future representation. FASTER-WAM invokes the video expert once, stores its per-layer keys and values, and serves that context to every action step (Zhao et al., 2026a); related work further distills, compresses, or reuses this future context (Li et al., 2026a; Zhang et al., 2026a; Huang et al., 2026; Wang et al., 2026a). This strategy is effective: relative to joint video–action denoising it reduces latency from 588.2 ms to 195.29 ms and preserves LIBERO-Plus OOD success, from 49.1% to 73.6%. Yet it optimizes the wrong recurring term: once the future context exists, the action solver dominates the runtime (84% of it, 165.27 of 195.29 ms; Table 4, Figure 1). Existing cache designs do not directly address this object: visual methods refresh spatial latents or blocks using global rules (Ma et al., 2024b; Chen et al., 2024; Selvaraju et al., 2024;

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.



(a) Closed-loop success rate **(b) Paired inference latency**

Figure 1: The action solver is the object worth caching. (a) Closed-loop success of FASTER-WAM (charcoal) and HAC (teal) on LIBERO, RoboTwin 2.0, and matched LIBERO-Plus OOD. (b) Paired latency with the future context cached: HAC nearly halves solver cost ($1.63\times$ overall, $1.86\times$ action). Seed-42 in-distribution results at η=8 (or the matched OOD set); the headline η=12 runs in Tables 1–4 use four evaluations and reach $1.63\times/1.86\times$.

Zou et al., 2025; Liu et al., 2024), whereas action-policy methods reuse whole chunks, cross-chunk residuals, or offline/learned schedules (Oi et al., 2026; Zhao et al., 2026b; Ji et al., 2026b; Wang et al., 2026b; Ji et al., 2026a). These approaches are valuable, but they leave open whether a WAM can cache the action solver itself without treating every chunk position alike.

The unresolved design question. *The goal is to spend computation only where an action trajectory bends, while leaving the easy majority untouched.*

Our answer starts from the structure of the output field. The future context is read-only after its single video pass, while the 32 positions in an action chunk follow visibly different denoising paths: most move nearly linearly, but a small set associated with contact and gripper transitions bends sharply. Uniform reuse wastes computation on the easy majority; uniform extrapolation risks the hard minority. HAC therefore treats each chunk position as a token, estimates scale-free curvature from the three most recent FULL fields, fills skipped steps by copy, first-order extrapolation, or a smoothstep-damped update, and uses a curvature-normalized drift on chaotic positions to decide when to invoke the solver again. The wrapper caches the complete action output field rather than internal features, adds no learned parameters, and leaves the future K/V cache untouched.

Our contributions are:

1. ***The action solver as a caching object.*** We show that the action solver, not the future context, is the dominant remaining cost of a WAM, and that its redundancy is ***long-tailed*** and ***physically interpretable*** across chunk positions—a regime that neither visual-token caches nor block-feature action caches are built for.
2. **HAC.** A training-free, online, **curvature-guided heterogeneous caching** rule for the action solver, plus a leaner variant that also skips the final refresh. HAC cuts solver evaluations from 10 to 4 (a 60% reduction) and delivers $1.63\times$ overall and $1.86\times$ action speedup over FASTER-WAM.
3. ***Quality-neutral evidence at scale.*** Across 3,000 RoboTwin episodes and 1,200 LIBERO episodes per run, and the 9,305 matched LIBERO-Plus OOD tasks, HAC moves success by $+0.09$ points on RoboTwin seed 42, $+0.35$ points on LIBERO seed 42, and -0.22 points at the conservative η=8 budget on the matched OOD set—**within noise**.

2 RELATED WORK

2.1 FUTURE-CONDITIONED ACTION GENERATION

World Action Models (WAMs) sit at the intersection of predictive video modeling and action generation. Earlier visuomotor policies either discretized control with a large vision–language backbone or predicted continuous action chunks directly (Brohan et al., 2023; Zitkovich et al., 2023; Kim

et al., 2024; Ghosh et al., 2024; Black et al., 2024). Recent work makes the efficiency target explicit: TinyVLA reduces the backbone and action representation (Wen et al., 2025), RoboLLM grounds language models in robot vision (Long et al., 2024), and RSS studies analyze VLA speed–success and spatial-representation trade-offs (Kim et al., 2025; Qu et al., 2025). Layer skipping and compact adapters push this line further (Zhang et al., 2026b; Wang et al., 2026c), while MeanFlow collapses policy denoising to one step (Sheng et al., 2026). These methods change the backbone, representation, or number of solver steps. Our setting is orthogonal: the released Faster-WAM backbone and its ten-step flow solver are fixed, and only redundant evaluations of the already-conditioned action field are replaced.

The predictive-future branch is complementary. Joint-WAM-style models couple video and action denoising, whereas Fast-WAM removes test-time future slots and Faster-WAM restores them with sparse mixed attention and a fused video K/V cache (Yuan et al., 2026; Zhao et al., 2026a; Bi et al., 2025; Li et al., 2026b). The audited implementation invokes the video expert once, fuses K/V at layers $\{0, 4, 8, 12, 16, 20, 24, 28\}$, and reuses it for all ten action steps. HAC targets a different object from future construction: the action-expert output field $v_\theta(A, \tau)$, not the future K/V state.

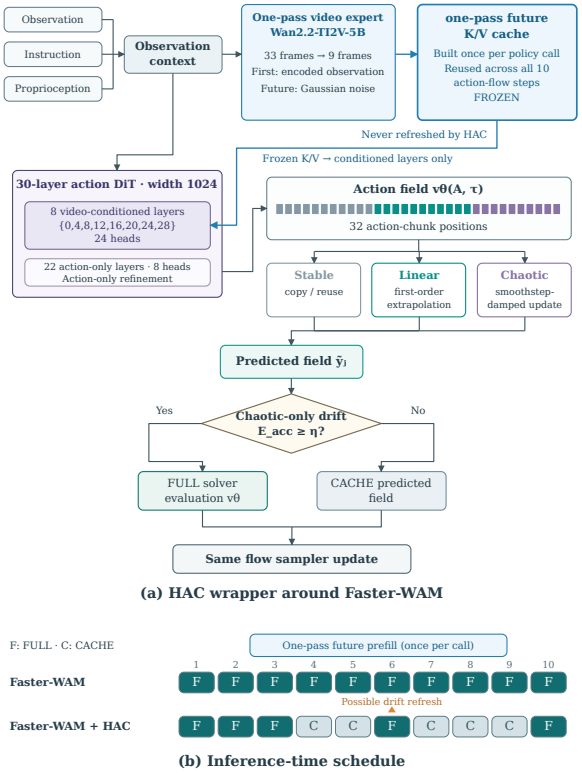


Figure 2: HAC caches the solver, not the future. (a) A frozen future K/V cache feeds a solver that scores the 32 chunk positions by curvature from three FULL anchors and fills skipped steps by reuse, extrapolation, or smoothstep damping, gated by chaotic drift. (b) Faster-WAM uses ten dense steps; HAC five at η=8 and four at η=12, at equal success.

2.2 Caching and Adaptive Computation in Diffusion and Flow Models

The closest efficiency literature can be organized by its optimized object. Layer/block caches reuse internal U-Net or DiT features on fixed or learned schedules (Ma et al., 2024b;a; Li et al., 2024); token caches select a subset of visual tokens using similarity or importance (Zou et al., 2025; Zhang et al., 2025); and trajectory caches forecast intermediate features and calibrate the forecast error (Kahatapitiya et al., 2025; Qiu et al., 2025; Chu et al., 2025; Zheng et al., 2026; Shen et al., 2025; Cai et al., 2025). Their common assumption is that the cached state is an internal feature map of an image or video generator. The refresh signal is consequently a feature residual, attention score, or global trajectory probe.

HAC differs at the level of the cached mathematical object. Each action-chunk position is a token in $\mathbb{R}^{d_a}$, and the cached value is the complete solver output at that token. Curvature is estimated from three FULL field outputs indexed by the actual solver step, the stable/linear/chaotic partition is recomputed only after a new anchor, and the chaotic-only drift is accumulated until it reaches η. Thus the criterion is online, token-specific, and output-space; it requires neither an offline schedule nor a probe network. The contrast matters because a visual-token cache can be accurate while the action field is changing sharply at contact or gripper transitions, and conversely a global threshold can waste compute on the easy majority of action positions.

Several 2025–2026 systems make adaptive caching more global or more predictive (Kahatapitiya et al., 2025; Qiu et al., 2025; Chu et al., 2025; Zheng et al., 2026), but none preserves the future con-

text while selectively recomputing a physical action position. The relevant unit for WAM inference is thus the heterogeneous action field, not a visual feature block.

2.3 Efficient VLA and world-model evaluation

Efficiency also appears in the data and evaluation layer: Human2Robot learns actions from paired human–robot videos, and recent VLA work studies spatial representations and speed–success calibration under realistic manipulation protocols (Xie et al., 2026; Qu et al., 2025; Kim et al., 2025). These works motivate evaluating closed-loop success and distribution shift rather than a decoder FLOP count alone, which we do for LIBERO, LIBERO-Plus, and RoboTwin 2.0. HAC is not a new VLA backbone, a learned controller, a cross-episode retrieval system, or a future K/V manager; it is a training-free wrapper around an existing flow-matching solver, and this object-level distinction is what lets us test quality neutrality on the same checkpoint.

3 Preliminaries: the Action Solver as the Bottleneck

At control step t, a WAM receives an image o_t, a language instruction l, and proprioception s_t, and emits an action chunk $\hat{A}_t \in \mathbb{R}^{H\times d_a}$. Here H=32: 33 observation frames are decimated by a factor of 4 into 9 video frames, which yields 32 action positions. The action dimension is d_a=7 on LIBERO (end-effector pose plus gripper) and d_a=14 on RoboTwin (bimanual). Runtime proceeds in two phases. In the first, the video expert (Wan2.2-TI2V-5B; 30 blocks, width 3072) is invoked a single time, at the highest video-flow noise τ_v=1, over the clean current-frame latent together with Gaussian future slots. It returns a per-layer key/value hierarchy, which a depth-wise fusion (`interval_weighted_sum`, seeded from the current layer) folds into a compact action-side context $\widehat{C}_t = \{\widehat{K}_j, \widehat{V}_j\}_{j\in\mathcal{J}}$ at the conditioning stages $\mathcal{J}$ (M=8 of the L=30 action stages: $\{0, 4, 8, 12, 16, 20, 24, 28\}$). This context is built once and then read by all later steps. In the second phase, the action expert—a 30-layer flow-matching DiT of width 1024—runs K=10 denoising steps; at step k it emits a velocity field

$$v_\theta^{(k)} \;=\; v_\theta\Big(A_t^{(k)}, \tau_a^{(k)} \mid s_t, l, \widehat{C}_t\Big) \in \mathbb{R}^{H\times d_a}, \qquad A_t^{(k+1)} = A_t^{(k)} + \Delta_k\, v_\theta^{(k)}, \tag{1}$$

where Δ_k is the scheduler-provided step delta and the implementation advances exactly as `sample` + `model_output` $\times\, \Delta_k$ (Lipman et al., 2023). Only $A_t^{(k)}$ changes, and only the solver is executed over and over. Throughout, we call a real evaluation of v_θ a *FULL* step and a predicted stand-in a *CACHE* step. The released model is trained with a joint flow-matching objective (masked per-token MSE on video and action targets with $\lambda_{\text{video}} = \lambda_{\text{action}} = 1$), and HAC retrains nothing. Inference runs in BF16 on the same 10-step schedule for all compared methods; solver state is cleared at every policy call and never crosses an observation or replan boundary (10 steps on LIBERO, 28 on RoboTwin).

Table 4 and §5.3 locate the cost: once the future context exists, the solver alone takes 84% of the runtime. That is the term HAC attacks.

4 Method

HAC is a wrapper rather than a new backbone. Figure 2 separates the two objects involved: the one-pass future K/V context is built once and then left frozen, while the action solver is evaluated selectively, position by position, under a curvature signal read from its own recent output fields. The three subsections below define the difficulty score, the per-token predictor for skipped steps, and the recomputation trigger.

4.1 Difficulty as curvature along the denoising path

Take the three most recent FULL outputs of the solver, written $\{(y_{j_2}, j_2), (y_{j_1}, j_1), (y_{j_0}, j_0)\}$ with $j_2 > j_1 > j_0$ in denoising order. Regarding each of the H chunk positions as a token of dimension d_a, finite differences between consecutive outputs give two velocities and one acceleration,

$$\mathbf{v}_{j_2,i} = \frac{y_{j_2,i} - y_{j_1,i}}{j_2 - j_1}, \quad \mathbf{v}_{j_1,i} = \frac{y_{j_1,i} - y_{j_0,i}}{j_1 - j_0}, \quad \mathbf{a}_{j_2,i} = \frac{\mathbf{v}_{j_2,i} - \mathbf{v}_{j_1,i}}{j_2 - j_1}, \tag{2}$$

from which we form a scale-free curvature

$$\kappa_i = \frac{\|\mathbf{a}_{j_2,i}\|_2}{\|\mathbf{v}_{j_2,i}\|_2^2 + \varepsilon}, \qquad \varepsilon = 10^{-8}. \tag{3}$$

This finite-difference curvature follows (Feng et al., 2026). We read κ_i as a normalized turning score: it stays small while a position glides smoothly down the denoising path and grows when the path bends sharply. Dividing by the local speed is what makes one threshold usable everywhere, across tokens and across action dimensions whose scales differ. Slicing the curvature distribution at two quantiles (p_s, p_c) yields three token sets,

$$\mathcal{I}_{\text{stable}} = \{i : \kappa_i < Q_{p_s}(\kappa)\}, \quad \mathcal{I}_{\text{chaotic}} = \{i : \kappa_i \geq Q_{p_c}(\kappa)\}, \quad \mathcal{I}_{\text{linear}} = \text{rest}, \tag{4}$$

with p_s=0.30 and p_c=0.70. Because these sets are a function of the three newest FULL anchors, they can only change when a new anchor arrives; before three anchors exist the solver simply runs in full.

Figure 3 makes the partition concrete: the same solver produces near-stationary positions that can be copied, nearly straight trajectories that can be extrapolated, and a minority that turns sharply and must be damped. A single global reuse rule cannot serve all three at once.

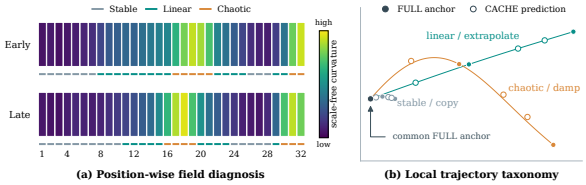


Figure 3: Action positions have heterogeneous solver dynamics: under the same solver, positions are near-stationary (copy), nearly linear (extrapolate), or turning (damped). Groups refresh only at a new FULL anchor; the trajectories are conceptual.

4.2 Filling skipped steps, token by token

Let $j^\star = j_2$ mark the latest FULL step and let $\delta = j - j^\star$ count solver steps since then. A skipped step is reconstructed position-wise:

$$\tilde{y}_{j,i} = \begin{cases} y_{j^\star,i}, & i \in \mathcal{I}_{\text{stable}}, \\ y_{j^\star,i} + \delta\, \mathbf{v}_{j_2,i}, & i \in \mathcal{I}_{\text{linear}}, \\ y_{j^\star,i} + \delta\, \mathbf{v}_i^{\text{adapt}}(\delta), & i \in \mathcal{I}_{\text{chaotic}}. \end{cases} \tag{5}$$

Stable positions are simply carried over. Linear positions continue along the latest velocity. Chaotic positions, for which a first-order step would overshoot, instead use a curvature-aware damping that mixes the two newest FULL velocities through a cubic smoothstep (Feng et al., 2026),

$$\mathbf{v}_i^{\text{adapt}}(\delta) = (1 - \alpha_\delta)\mathbf{v}_{j_2,i} + \alpha_\delta\, \mathbf{v}_{j_1,i}, \qquad \alpha_\delta = 3x_\delta^2 - 2x_\delta^3, \quad x_\delta = \min\Big(\frac{\delta}{n_{\max}}, 1\Big). \tag{6}$$

As the run moves further past $j^\star$, $\alpha_\delta \to 1$ and the update turns progressively more conservative, which keeps error on contact positions from piling up during an extended cache streak.

4.3 A chaotic-token budget for recomputation

HAC still has to decide when to pay for a real evaluation. Its trigger is a scale-free drift measured on chaotic tokens only (Feng et al., 2026). Writing $\tilde{y}_j$ for the current stand-in and $\tilde{y}_{j-1}$ for the immediately preceding field (a stand-in, or the latest FULL output at the first skipped step after an anchor),

$$e_i(j) = \kappa_i \left\|\tilde{y}_{j,i} - \tilde{y}_{j-1,i}\right\|_2, \quad i \in \mathcal{I}_{\text{chaotic}}, \qquad E(j) = \frac{1}{|\mathcal{I}_{\text{chaotic}}|} \sum_{i \in \mathcal{I}_{\text{chaotic}}} e_i(j). \tag{7}$$

Multiplying the deviation by κ_i strips out global scale, so one budget η serves all tokens and steps. Consecutive skipped steps add to a running sum, $E_{\text{acc}} \leftarrow E_{\text{acc}} + E(j)$, and once $E_{\text{acc}} \geq \eta$ the next step is forced to FULL. Because only chaotic tokens vote, harmless motion in the easy majority never triggers a recompute. Two structural guards complete the design: the opening three steps (j=0, 1, 2, zero-indexed) always run FULL to seed the curvature history, and a cache streak is capped at $n_{\max}$=6 before a FULL is imposed. The default HAC (code mode `wc-adapt`) also runs the last step FULL; HAC-FAST (`wc-adapt-no-last-full`) drops it. Defaults are p_s=0.30, p_c=0.70,

Algorithm 1 HAC: caching the action solver with curvature-based token groups.

Require: flow field v_θ, horizon K, budget η, quantiles (p_s, p_c), streak cap $n_{\max}$, whether to force the last step.

let anchor buffer $\mathcal{H} \leftarrow \emptyset$, risk sum $E_{\text{acc}} \leftarrow 0$, streak $c \leftarrow 0$
for each solver step $j = 0, \ldots, K-1$ **do**
  if $\mathcal{H}$ holds fewer than 3 anchors **or** ($j = K-1$ and last step is forced) **or** $E_{\text{acc}} \geq \eta$ **or** $c \geq n_{\max}$ **then**
    run FULL: $y_j \leftarrow v_\theta(\cdot)$; push y_j into $\mathcal{H}$; refresh κ (Eq. 3) and the token sets (Eq. 4); reset $E_{\text{acc}} \leftarrow 0$ and $c \leftarrow 0$
  else
    run CACHE: advance $c \leftarrow c + 1$; build $\tilde{y}_j$ position-wise (Eqs. 5, 6)
    add the chaotic drift to E_{acc}
  end if
  advance the same flow sampler with the selected field y_j (FULL) or $\tilde{y}_j$ (CACHE)
end for

$n_{\max}$=6, η=1.0; *the in-distribution and cross-model headline runs use* η=12*, the conservative matched LIBERO-Plus OOD run* η=8 (§5.4). Algorithm 1 sketches the wrapper, whose per-call overhead—two quantiles and one norm per token—is negligible next to a solver evaluation.

A remark on the frozen future. We also experimented with refreshing the cached future K/V using action-side signal. Though the K/V values shifted appreciably, the resulting action moved by less than 1% and task success did not improve, so we keep $\widehat{C}_t$ frozen and cache only the solver.

5 Experiments

Protocol. All runs load the released FASTER-WAM checkpoint (video expert from Wan2.2-TI2V-5B; 30 action layers, width 1024, head dimension 128, M=8 of L=30 conditioning stages, horizon 32, 9 video frames) and evaluate on RoboTwin 2.0 (clean and randomized) (Chen et al., 2025), LIBERO (Liu et al., 2023), and the seven-shift LIBERO-Plus suite (Fei et al., 2025). RoboTwin spans 50 tasks with 30 clean and 30 randomized episodes each (3,000 episodes), replanning every 28 steps; LIBERO uses four suites (40 tasks, 30 trials, 1,200 episodes), replanning every 10 steps; LIBERO-Plus contributes one trial per task. All methods share the 10-step schedule, BF16 precision, and the released inference configuration (flow shift 5.0, cameras assembled into one observation, proprioception in the context, language context encoded once per instruction), with seed coverage reported per table. HAC adds neither training nor parameters. Latency follows a shared protocol (batch 1, 224×448, 5 warm-up plus 10 synchronized iterations) that always pits baseline and HAC in the *same* process on the same GPU.

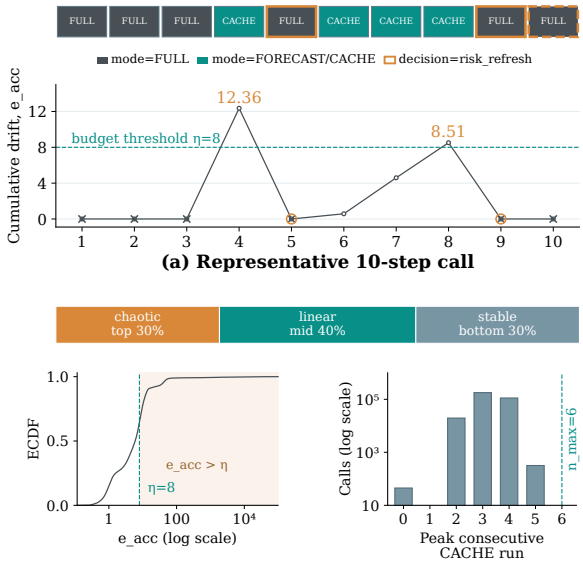


Figure 4: A small chaotic minority determines recomputation. (a) Representative call: chaotic-token drift crossing the budget triggers the next FULL step (dashed: forced closing FULL). (b) Aggregate evidence: per-step group quota (top), cumulative-drift distribution (left), longest CACHE streak (right).

Table 1: Success rates (%) on LIBERO (four suites, 40 tasks). The upper block is reported baselines; the lower block is our reproductions (seed 42). Gold marks our best average and blue shading our paired baseline.

Method	P.T.	Spa.	Obj.	Goa.	Lon.	Avg.
$\pi_{0.5}$ (Intelligence et al., 2025)	✓	98.8	98.2	98.0	92.4	96.9
LingBot-VA (Li et al., 2026b)	✓	98.5	99.6	97.2	98.5	98.5
Motus (Bi et al., 2025)	✓	96.8	99.8	96.6	97.6	97.7
Fast-WAM (Yuan et al., 2026)	×	98.2	100.0	97.0	95.2	97.6
Joint-WAM (Yuan et al., 2026)	×	99.6	99.4	98.2	96.8	98.5
FASTER-WAM (Zhao et al., 2026a)	×	99.6	99.8	98.2	98.2	99.0
Our reproductions (released checkpoints, no embodied pretraining)						
FASTER-WAM (full solver)	×	98.80	100.00	98.60	97.20	98.65
FASTER-WAM + HAC	×	99.00	100.00	99.00	98.00	**99.00**
FASTER-WAM + HAC-FAST	×	98.00	99.67	99.33	98.33	98.83
Fast-WAM (full solver)	×	96.80	99.20	97.60	91.60	96.30
Fast-WAM + HAC	×	97.60	100.00	96.80	93.60	97.00

Table 2: Success rates (%) on RoboTwin 2.0 (50 tasks). The upper block is reported baselines; the lower block is our reproductions (seed 42). Gold marks our best average and blue shading our paired baseline.

Method	P.T.	Clean	Rand.	Avg.
$\pi_{0.5}$ (Intelligence et al., 2025)	✓	82.7	76.8	79.8
Motus (Bi et al., 2025)	✓	88.7	87.0	87.9
LingBot-VA (Li et al., 2026b)	✓	92.9	91.5	92.2
Fast-WAM (Yuan et al., 2026)	×	91.9	91.8	91.9
Joint-WAM (Yuan et al., 2026)	×	90.8	90.3	90.6
FASTER-WAM (Zhao et al., 2026a)	×	92.8	92.3	92.6
Our reproductions (released checkpoints, no embodied pretraining)				
FASTER-WAM (full solver)	×	92.16	92.46	92.31
FASTER-WAM + HAC	×	93.00	91.80	**92.40**
FASTER-WAM + HAC-FAST	×	91.00	89.20	90.10
Fast-WAM (full solver)	×	91.40	89.93	90.67
Fast-WAM + HAC	×	91.47	90.33	90.90

5.1 ACCURACY INSIDE THE TRAINING DISTRIBUTION

In both accuracy tables, the upper block is as reported by Zhao et al. (2026a) (their Tables 1–2; 50 rollouts per LIBERO task, 100 trials per RoboTwin setting); the lower block is our reproductions under the shared protocol of §5 without embodied pretraining (P.T.). Rows name the frozen backbone followed by the solver policy, so the same HAC wrapper is read as a policy: the Faster-WAM rows wrap the released Faster-WAM checkpoint and the Fast-WAM rows the released Fast-WAM checkpoint, which has no test-time future path. At seed 42 and η=12, HAC reaches 99.00% on LIBERO against 98.65% for the matched FASTER-WAM baseline and 92.40% on RoboTwin against 92.31%; HAC-FAST reaches 98.83% and 90.10%, and no suite shows a monotone degradation. The closing FULL step is therefore a cheap but real quality guard at η=12: dropping it costs 2.30 points on RoboTwin and 0.17 on LIBERO for only 2.16 ms, and at η=8 it costs only 0.35 points on RoboTwin and is neutral on LIBERO (Figure 5b). Both tables also carry the Fast-WAM block from §5.5.

5.2 BEHAVIOR UNDER DISTRIBUTION SHIFT

Table 3 probes robustness on LIBERO-Plus over seven shifts, restricted to the 9,305 tasks completed by both arms. HAC averages 73.61% against 73.82% for our FASTER-WAM baseline (−0.22 points). The per-shift deltas span from negative on Noise to positive on Light, with binomial standard errors of roughly 1.2–1.5 points, so the spread does not establish a systematic loss of the OOD benefit (per-shift breakdown in Figure A1). This run uses the conservative η=8; since latency is monotone in the budget (Figure 5), its acceleration is a lower bound and we do not claim OOD quality at the headline budget. P.T. denotes embodied pretraining.

Table 3: Success rates (%) on LIBERO-Plus across seven distribution shifts. The upper block is reported baselines on the full suite; the lower block is our matched 9,305-task intersection, where Δ is HAC minus our FASTER-WAM reproduction. Gold marks the best/second-best average (bold/underline), red text a positive change, and blue shading the baseline or a non-positive change.

Method	P.T.	Camera	Robot	Lang.	Light	Backg.	Noise	Layout	Avg.	Δ
Reported by Zhao et al. (2026a) (full suite)										
UniVLA (Bu et al., 2025)	✓	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9	—
OpenVLA-OFT (Kim et al., 2025)	✓	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6	—
π_0 (Black et al., 2024)	✓	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6	—
π_0-Fast (Pertsch et al., 2025)	✓	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6	—
WorldVLA (Cen et al., 2025)	✓	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0	—
Fast-WAM (Yuan et al., 2026)	×	18.8	45.7	70.1	83.2	45.7	29.8	62.7	49.1	—
Joint-WAM (Yuan et al., 2026)	×	37.5	64.5	93.0	95.0	55.9	47.3	79.6	66.3	—
FASTER-WAM (Zhao et al., 2026a)	×	53.8	71.6	94.7	96.3	61.3	63.6	79.1	73.6	—
Ours, matched 9,305-task intersection										
FASTER-WAM (full solver)	×	55.16	69.42	93.56	98.16	60.69	69.29	78.73	**73.82**	—
FASTER-WAM + HAC	×	55.16	69.23	93.69	98.62	60.59	67.44	79.14	<u>73.61</u>	−0.22

Table 4: Component-wise inference latency (ms). The upper block is the reported profile (NVIDIA L20); the lower block is our paired protocol (NVIDIA RTX PRO 6000 Blackwell). Joint-WAM denoises video and action jointly, so its visual and action branches cannot be separated (dashes). Gold marks the best latency, red text a reduction, and blue shading the paired baseline.

Model	VAE Enc.	Visual	Action ↓	Overall ↓
Reported by Zhao et al. (2026a) (NVIDIA L20)				
Joint-WAM (Yuan et al., 2026)	10.47	—	—	559.84
Fast-WAM (Yuan et al., 2026)	10.47	27.67	276.56	320.97
FASTER-WAM (Zhao et al., 2026a)	10.55	43.04	192.11	252.95
Ours, paired protocol (NVIDIA RTX PRO 6000 Blackwell)				
Joint-WAM	6.49	—	—	588.20
Fast-WAM (full solver)	6.10	29.03	294.70	334.30
FASTER-WAM (full solver)	6.04	25.60	165.27	195.29
FASTER-WAM + HAC	6.10	25.81	74.15	107.49
FASTER-WAM + HAC-FAST	6.14	25.67	72.42	105.33
Fast-WAM + HAC	6.35	25.91	141.08	173.82

5.3 WHERE THE TIME GOES

Table 4 breaks down inference latency: once the future context exists, the solver consumes 165.27 ms (84%) of the 195.29 ms total, against 25.60 ms for the visual branch and 6.04 ms for VAE encoding. Because the future context stays frozen, HAC removes this redundancy without weakening conditioning, dropping solver latency to 74.15 ms (1.63× overall, 1.86× on the solver, measured in one process). The upper block reproduces the reference profile of Zhao et al. (2026a); the lower block uses our paired protocol (BF16, batch 1, 224×448, horizon 32, 10 denoising steps, 5 warm-up plus 10 GPU-synchronized measurements), where all Faster-WAM and Fast-WAM rows share the η=12 operating point and Joint-WAM is the separate reference. The table also carries the Fast-WAM + HAC row from §5.5, measured under the same paired protocol.

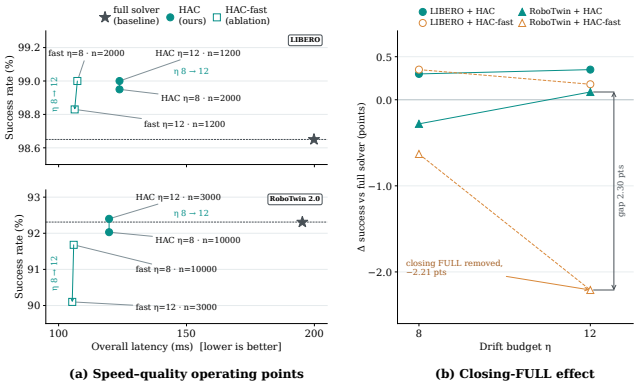
5.4 DESIGN CHOICES AND SENSITIVITY

Appendix Table A2 summarizes the sensitivity study. Part (a) varies the cache policy over field evaluations per call and latency; the plain ten-step solver (`full10`) is the reference and both η=12 wrappers use four evaluations. HAC improves the seed-42 averages by 0.09 points on RoboTwin and 0.35 on LIBERO. HAC-FAST reaches the same four evaluations by dropping the closing FULL step, but its marginal latency gain (105.33 vs 107.49 ms) costs 2.3 points on RoboTwin, so we treat it as a diagnostic rather than a default. Part (b) lists the architecture-side baselines reported by Zhao et al. (2026a); HAC leaves that branch and the interaction graph fixed, so this panel is orthogonal to the cache. The drift budget η is the remaining knob, and Figure 5 characterizes it at

scale. Moving from η=8 to η=12 reduces overall latency by 12.1 ms on LIBERO (123.81 → 111.72) and 12.3 ms on RoboTwin (119.74 → 107.49) while success changes by +0.05 and +0.37 points—within noise—so latency is monotone in the budget and in-distribution quality is flat. We therefore headline η=12, which matches the ten-step solver at the fewest evaluations (4); the conservative η=8 uses five and gives 98.95/92.03. The HAC–HAC-FAST gap widens from −0.05 to +0.17 points on LIBERO and from +0.35 to +2.30 on RoboTwin as η rises from 8 to 12: the closing FULL step is nearly free at a conservative budget and costly at an aggressive one, exactly as a drift guard charged with a longer skipped horizon should behave. We fix one operating point per protocol instead of tuning η per task. The partition is what carries the benefit: earlier diffusion caches either reuse every token or apply a single forecast rule, whereas on action fields blanket reuse trails on contact positions and blanket extrapolation overshoots stable ones.

5.5 Generality across world action models

HAC changes only the action-denoising loop, not the video branch, so the same wrapper applies to a WAM that never imagines a future at test time. We port it without tuning to Fast-WAM (Yuan et al., 2026), whose action expert attends only to the current-frame latent and has no future K/V cache; curvature, partition, predictor, and drift budget are kept identical (p_s=0.30, p_c=0.70, $n_{\max}$=6, η=12). The matched native-versus-cached comparison is folded into the tables that carry the corresponding claim (accuracy in Tables 1 and 2, latency in Table 4): HAC raises Fast-WAM's LIBERO average from 96.30% to 97.00% and its RoboTwin average from 90.67% to 90.90%, while cutting its solver latency from 253.44 to 141.08 ms. This locates the cached object: because the same schedule transfers, the redundancy belongs to the action solver rather than to Faster-WAM's particular future path.



(a) Speed-quality operating points (b) Closing-FULL effect

Figure 5: The drift budget η as a single speed–quality knob. (a) Latency–success operating points for LIBERO (circles) and RoboTwin 2.0 (triangles): filled markers HAC, hollow HAC-FAST, stars the ten-step solver; arrows run from η=8 to η=12. (b) Success change versus the full solver; the widening HAC–HAC-FAST gap shows the closing FULL step guards longer skipped horizons (2.21 points on RoboTwin at η=12; LIBERO-Plus was not run at η=12).

6 Limitations and Conclusion

HAC caches at the output level and, being training-free, only lowers the *number* of solver evaluations, not the cost of each, so its speedup is bounded by the fraction of steps it can skip (at most K). Our OOD evidence rests on a matched 9,305-task LIBERO-Plus subset rather than the full 10,030; the full-suite baseline (74.30%) agrees with those matched numbers. The quantile settings (p_s=0.30, p_c=0.70) and the predictor are fixed across benchmarks, and η is a single speed–quality knob rather than a per-task hyperparameter (Figure 5). We restrict scope to the solver: we tried refreshing the future cache and set it aside, though a learned scheme for the future side is not ruled out.

Conclusion. After future conditioning is computed once, a WAM's bottleneck is its action solver, and that solver's redundancy is heterogeneous across chunk positions. HAC exploits both facts with a simple, training-free output-field wrapper that cuts solver evaluations from 10 to 4 at no measurable quality cost.

AI USE STATEMENT

In this work, we used generative AI tools to draft and edit the manuscript, to implement and debug analysis code, and to design experiments and interpret results. We did not use generative AI tools for the other disclosure-required tasks (e.g., generating synthetic data, formulating mathematical claims, and developing theoretical models or hypotheses), which are not applicable to this work. We reviewed all AI-assisted work: we checked AI-generated research ideas against the primary literature cited here, verified every reported number against the raw logs and per-task result files produced by the released code, and re-ran the aggregation scripts that generate the paper tables. All experimental designs and result interpretations suggested by AI tools were reviewed and validated by the authors against the measured evidence. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies inference efficiency of robot manipulation policies in simulated benchmarks (LIBERO, LIBERO-Plus, RoboTwin 2.0) and does not involve human subjects or personal data. Faster policies at equal success could improve real-time robot control; we are not aware of specific dual-use concerns beyond those common to robot learning.

REPRODUCIBILITY STATEMENT

All methods are training-free and evaluated with released checkpoints. The cache wrapper is an isolated module, the launchers are per-experiment shell scripts, results are written incrementally with resume support, and every aggregate row is appended to a single experiment ledger. We will release the cache module, launchers, and the aggregation scripts that reproduce all tables.

REFERENCES

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.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023.

Qingwen Bu, Yanting Yang, Jiahan 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.

Yuan Cai et al. MagCache: Fast video generation with magnitude-aware cache. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jun Cen, Chaohui Yu, Haoran Yuan, Yansong Jiang, Shijie Huang, Jiaqi Guo, Xin Li, Yufeng Song, Hao Luo, Feng Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Pengtao Chen, Mengzhu Shen, Peng Ye, Jian Cao, Chong Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Δ-DiT: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024.

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.

Huanpeng Chu, Wei Wu, Guanyu Feng, and Yutao Zhang. Omnicache: A trajectory-oriented global perspective on training-free cache reuse for diffusion transformer models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. doi: 10.1109/ICCV51701.2025.01513.

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: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Weilun Feng, Guoxin Fan, Haotong Qin, Mingqiang Wu, Yuqi Li, Xiangqi Li, Zhulin An, Libo Huang, Dingrui Wang, Longlong Liao, Michele Magno, Yongjun Xu, and Chuanguang Yang. WorldCache: Accelerating world models for free via heterogeneous token caching. In *International Conference on Machine Learning (ICML)*, 2026.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

Jiakai Huang, Zhongbo Wu, Zheng Zhang, Zihan Wang, Shan You, and Tao Huang. Foresight without seeing: Latent futures for world action models. *arXiv preprint arXiv:2608.11605*, 2026.

Physical Intelligence et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Kangye Ji, Jianbo Zhou, Yuan Meng, Ye Li, Hanyun Cui, and Zhi Wang. Sparse actiongen: Accelerating diffusion policy with real-time pruning. *arXiv preprint arXiv:2601.12894*, 2026a.

Kangye Ji et al. Block-wise adaptive caching for accelerating diffusion policy. In *International Conference on Learning Representations (ICLR)*, 2026b.

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S. Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. doi: 10.1109/ICCV51701.2025.01414.

Moo Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

Jiajun Li, Tiecheng Guo, Yifan Ye, Rongyu Zhang, Xiaowei Chi, Qianpu Sun, Ying Li, Yunfan Lou, Yan Huang, Zhihe Lu, Meng Guo, and Shanghang Zhang. Efficient-WAM: A 1B-parameter world-action model with low-cost future imagination. *arXiv preprint arXiv:2606.10040*, 2026a.

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*, 2026b.

Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. doi: 10.1109/CVPR52733.2024.00686.

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 (ICLR)*, 2023.

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 (NeurIPS)*, 2023.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Raghav Pandey, and Hang Xu. Timestep embedding tells: It's time to cache for video diffusion model. *arXiv preprint arXiv:2411.19108*, 2024.

Zijun Long, George Killick, Richard McCreadie, and Gerardo Aragon-Camarasa. Robollm: Robotic vision tasks grounded on multimodal large language models. In *2024 IEEE International Conference on Robotics and Automation*, 2024. doi: 10.1109/ICRA57147.2024.10610797.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *Advances in Neural Information Processing Systems*, 2024a. doi: 10.52202/079017-4236.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

NVIDIA. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Ryuji Oi, Hikari Otsuka, Kosuke Matsushima, Yuki Ichikawa, Masato Motomura, Tatsuya Kaneko, and Daichi Fujiki. Actioncache: Training-free acceleration for vision-language-action models with action caching and refinement. *arXiv preprint arXiv:2607.06370*, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Francesco Nori, et al. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems (RSS)*, 2025.

Junxiang Qiu, Lin Liu, Shuo Wang, Jinda Lu, Kezhou Chen, and Yanbin Hao. Accelerating diffusion transformer via gradient-optimized cache. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. doi: 10.1109/ICCV51701.2025.01636.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.011.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. FORA: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.

Xuan Shen, Zhao Song, Yufa Zhou, Bo Chen, Yanyu Li, Yifan Gong, Kai Zhang, Hao Tan, Jason Kuen, Henghui Ding, Zhihao Shu, Wei Niu, Pu Zhao, Yanzhi Wang, and Jiuxiang Gu. Lazydit: Lazy learning for the acceleration of diffusion transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. doi: 10.1609/aaai.v39i19.34248.

Juyi Sheng, Ziyi Wang, Peiming Li, and Mengyuan Liu. Mp1: Meanflow tames policy learning in 1-step for robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. doi: 10.1609/aaai.v40i22.38919.

Jiayu Wang, Bin Zhu, Yue Yu, and Jingjing Chen. MoWAM: Explicit future motion prediction for efficient world action models. *arXiv preprint arXiv:2609.20709*, 2026a.

Siying Wang, Kangye Ji, Di Wang, and Fei Cheng. Evolving cache schedules for fast diffusion policy inference. *arXiv preprint arXiv:2607.20293*, 2026b.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026c. doi: 10.1609/aaai.v40i22.38931.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhi Xu, Ning Cheng, Chengjie Shen, Yaxin Peng, Feifei Feng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. In *IEEE Robotics and Automation Letters (RA-L)*, 2025.

Sicheng Xie, Haidong Cao, Zejia Weng, Zhen Xing, Haoran Chen, Shiwei Shen, Jiaqi Leng, Zuxuan Wu, and Yu-Gang Jiang. Human2robot: Learning robot actions from paired human-robot videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. doi: 10.1609/aaai.v40i13.38086.

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.

Chushan Zhang, Jinguang Tong, Xuesong Li, Yikai Wang, and Hongdong Li. Keep the future, drop the rollout: RIFT for world action models. *arXiv preprint arXiv:2608.11521*, 2026a.

Evelyn Zhang, Jiayi Tang, Xuefei Ning, and Linfeng Zhang. Training-free and hardware-friendly acceleration for diffusion models via similarity-based token pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. doi: 10.1609/aaai.v39i9.33071.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026b. doi: 10.1609/aaai.v40i22.38945.

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-wam: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026a.

Weisen Zhao, Lam Nguyen, Zhicong Lu, and Yuzhang Shang. C3ache: Accelerating world action models with cross inference chunk cache. *arXiv preprint arXiv:2606.08962*, 2026b.

Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ode for efficient diffusion transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. doi: 10.1609/aaai.v40i16.38349.

Brianna Zitkovich, Tianhe Yu, Sicheng 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 *Conference on Robot Learning (CoRL)*, 2023.

Chang Zou, Xinyin Liu, Ting Liu, Shichen Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *International Conference on Learning Representations (ICLR)*, 2025.

A Appendix

A.1 The curvature and drift, in full

We adopt the discrete curvature of WorldCache without modification. With FULL outputs $y_{j_0}, y_{j_1}, y_{j_2}$ ordered along denoising, the velocities and acceleration are those of the main text, giving $\kappa_i = \|\mathbf{a}_{j_2,i}\|_2/(\|\mathbf{v}_{j_2,i}\|_2^2 + \varepsilon)$ with ε=10^{-8}. The p_s=0.30 and p_c=0.70 quantiles of $\{\kappa_i\}_{i=1}^{H}$ define the three sets; since they depend only on the three newest FULL anchors, they change solely when an anchor is appended. The chaotic drift is $e_i(j) = \kappa_i\|\tilde{y}_{j,i} - \tilde{y}_{j-1,i}\|_2$ with $E(j) = |\mathcal{I}_{\text{chaotic}}|^{-1}\sum_{i\in\mathcal{I}_{\text{chaotic}}} e_i(j)$, summed until $E_{\text{acc}} \geq \eta$.

A.2 Wrapping an existing solver

HAC swaps the plain ten-iteration loop of the released Faster-WAM inference entry point for a planner that (i) evaluates the field, (ii) records it as a FULL anchor, (iii) on skipped steps forecasts the field without touching the action expert, and (iv) hands the stand-in to the same sampler step. The future K/V cache is never modified. Because the stand-in matches v_θ in shape and dtype, the same planner can wrap a compatible flow-matching action head; the released implementation is instantiated for Faster-WAM. Should a prediction be non-finite, the scheduler spacing invalid, or history missing, the planner falls back to a FULL evaluation. Solver state is cleared at each

Table A1: Positioning of HAC against the closest acceleration families. The second column is the object whose computation is skipped; the third is the refresh rule. HAC is the only entry that caches the action *output field* at token granularity with an online criterion and no offline search or learned component. Neutral bands separate cache families, while the gold/red final row identifies our proposed object and its online criterion.

Method	Cached object	Refresh decision	Needs training/search
DeepCache	U-Net sub-network features	fixed interval	no
FORA / Δ-DiT	block attention/MLP or residual features	fixed interval	no
Learning-to-Cache	DiT layers	learned router (offline)	offline optimization
TGATE	converged attention outputs	scheduled interval	no
TeaCache / MagCache	block residual (probe)	calibrated global threshold	offline calibration
EasyCache / FasterCache	transformation vectors / CFG residuals	runtime-adaptive	no
TaylorSeer / HiCache	block features (polynomial forecast)	fixed interval	no
ToCa / DuCa	visual tokens	token importance / random	no
PAB	attention pyramid	fixed interval	no
WORLDCACHE	visual token field	chaotic curvature drift	no
BAC	action block features	offline DP schedule	offline search
EVO	action block×step lattice	evolutionary search	offline search + rollouts
SAG	action activations	learned observation-conditioned pruner	trained pruner
ActionCache	whole action chunks	retrieval hit/miss	cross-episode memory
C3ache	residuals across chunks	fixed refresh interval	no
VLA-Cache / EfficientVLA	VLM visual tokens / layers	attention- or heuristic-based	no
Spec-VLA	draft action tokens	speculative verification	draft model
HAC (**ours**)	**action output field, per chunk-token**	**online chaotic curvature drift**	**no**

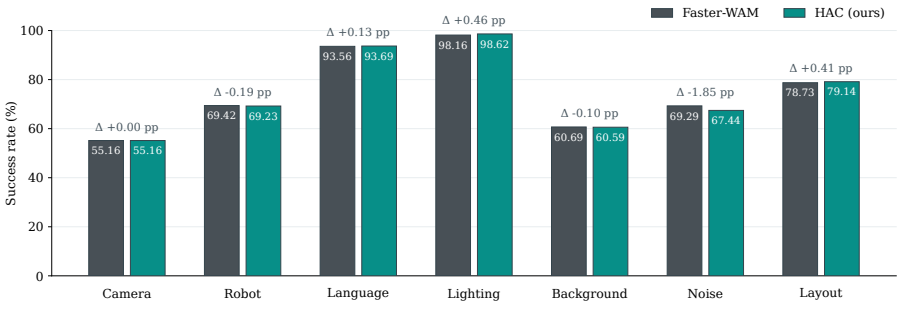


Figure A1: Out-of-distribution behavior on the matched 9,305 LIBERO-Plus tasks (one trial each, seed 42). Per-shift success of FASTER-WAM and HAC with deltas labeled; the largest drop is on Noise, the aggregate within binomial uncertainty.

policy call and never crosses an observation or replan boundary. Implementation mapping: HAC corresponds to `worldcache_mode=wc-adapt`, HAC-FAST to `wc-adapt-no-last-full`, and the plain ten-step baseline to `full10`.

A.3 ADDITIONAL ABLATIONS AND OOD PER-SHIFT BREAKDOWN

Table A2 reports the full cache-policy sweep and the architecture-side baselines, and Figure A1 gives the per-shift breakdown of the matched LIBERO-Plus comparison summarized in the main text.

A.4 CACHED-TRAJECTORY STATISTICS AND SAFEGUARDS

Anatomy of a cached trajectory. Over the LIBERO-Plus runs (which use η=8), HAC averages 5.29 FULL and 4.71 CACHE steps per call. Its most common schedule is three FULL steps, one mid-trajectory chaotic-drift refresh, and a closing FULL step (FFF · F · · · F), found in 37% of calls; the rest differ only in where the refresh falls. At the headline η=12 setting the same mechanism drops the closing step and averages four FULL steps. Quantile cuts keep the three sets balanced at 10/12/10 stable, linear, and chaotic tokens out of 32. At a typical step, then, about two thirds of the chunk can be reused or extrapolated and only one third calls for the damped predictor that gates recomputation.

Trigger statistics. Figure 4 aggregates 1,476,365 CACHE steps and 310,917 completed calls over LIBERO seeds 43 (experiments 15–16) and LIBERO-Plus (experiment 17). In the representative call

Table A2: Ablations. (a) Cache-policy ablation; (b) architecture-side baselines as reported by Zhao et al. (2026a). Arrows give the objective direction; gold marks the best entry in a block, red an improvement, and blue a baseline or decline.

(a) Cache policy

Policy	Eval./call ↓	RBT Avg. (%) ↑	LIB Avg. (%) ↑	Lat. (ms) ↓
FASTER-WAM (full solver)	10.00	92.31	98.65	195.29
FASTER-WAM + HAC	4.00	92.40	**99.00**	107.49
FASTER-WAM + HAC-FAST	4.00	90.10	98.83	**105.33**

(b) Faster-WAM architecture ablations (reported, LIBERO-Plus Avg. (%))

Variant	Avg.	Variant	Avg.
C-O Faster-WAM	51.00	Dense interaction	69.78
w/o KV-Fusion	69.99	Stride-2	71.65
w/o SparseMoT	69.78	Stride-4	73.57
Joint-WAM	66.27	Stride-7	71.05
Fast-WAM	49.14	Stride-14	70.05

of panel (a), crossing the budget η=8 at steps 4 and 8 triggers the next-step FULL evaluations at steps 5 and 9. Across the aggregate, 35.4% of recorded cumulative-drift values exceed the budget and the longest CACHE streak is five, so the $n_{\max}$=6 guardrail is never reached; quantile grouping fixes the per-step quota although group membership can change. Refresh is thus driven by the chaotic-token budget rather than a global mean or a generic fixed schedule.

Numerical safeguards. Curvature and drift are evaluated in FP32 on the field returned by the solver, and the token sets are updated only when a new FULL anchor is appended alongside the existing three. A fallback reruns a step in FULL whenever a prediction is non-finite, the anchor spacing is invalid, or history is missing; across the 11,201 cached calls logged in the LIBERO-Plus evaluation it never fired, which suggests the procedure is numerically well behaved in practice.

A.5 EVALUATION BUDGET

RoboTwin uses 30 clean and 30 randomized episodes per task at seed 42; LIBERO uses 30 trials per task at seed 42; LIBERO-Plus uses a single trial per task at seed 42. Latency is measured on one GPU, and the per-shift OOD figures aggregate the intersection of tasks finished by both the baseline and HAC.