GO FURTHER, BE LIGHTER: RECURRENT PREDICTIVE FUTURES FOR WORLD-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World-action models (WAMs) repeatedly compute predictive futures for control, incurring substantial inference cost. Asynchronous execution reduces waiting, but the overhead of temporal correction can limit effective action-generation throughput. Crucially, the predictive state conditioning the executing action already anticipates the outcome of that chunk before the robot reaches the next control boundary. Our *key insight* is that even imperfect predictions capture chunk-end changes in task-relevant regions. This creates an opportunity to advance the predictive future into a further future for subsequent control. To this end, we introduce the **Further-Future Propagator (FFP)**, which treats the predictive state as a recurrent state and learns a future-to-further-future transition that advances the predictive state across control rounds guided by factual evidence. Action relevance shapes how information important for action generation is preserved and updated, while a recurrent manipulation context modulates how the predictive state evolves across control rounds. The composed state conditions the frozen native action pathway and becomes the next recurrent input. Before each transition, a Confidence module estimates the marginal value of obtaining a fresh native prediction from the WAM, enabling adaptive refresh instead of a fixed refresh schedule. Across three architecturally distinct WAMs on CALVIN and RoboTwin 2.0, FFP improves control–compute trade-offs. On Faster-WAM/RoboTwin 2.0 Easy, FFP achieves $7.4\times$ the effective action-generation throughput of RTC while improving success rate from 76.1% to 92.2%. Across four real-world tasks with two WAMs, task-averaged completion time decreases by 34.3–39.3% relative to asynchronous baselines, with mean success rate maintained or improved.

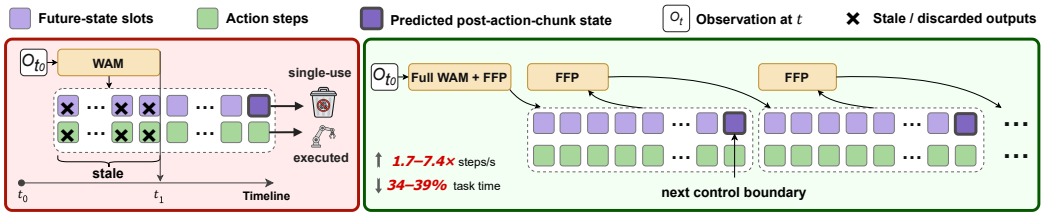
1 INTRODUCTION

World-action models (WAMs) extend vision-language-action policies (Brohan et al., 2023; Kim et al., 2024) by using predictions of future scene dynamics to guide action generation (Hu et al., 2025; Li et al., 2026). This predictive context provides look-ahead for control, but generating it adds inference latency that becomes waiting time between action chunks in a synchronous loop. To reduce these pauses, asynchronous execution computes the next chunk while the robot executes the current one (Black et al., 2025; Li et al., 2026). During this overlap, predictions anchored to an earlier observation may become available only after the robot has advanced, leaving part of the predicted horizon stale (Figure 1(a)). The returned action chunk must therefore remain temporally aligned with ongoing execution and compatible with actions already committed. Existing methods address this requirement through action-chunk alignment or feedback-grounded reconstruction of the post-action visual state (Black et al., 2025; Li et al., 2026).

Unfortunately, temporal consistency alone does not ensure efficient use of predictive computation. Real-Time Chunking (RTC) incurs guidance overhead and excludes from execution action steps whose scheduled times have elapsed during inference, while LingBot-VA's asynchronous pipeline adds a forward-dynamics pass before predicting the next video–action chunk (Black et al., 2025; Li et al., 2026). The resulting correction overhead and loss of usable action steps can limit effective action-generation throughput—usable action steps produced per unit of inference time.

A key opportunity is that the predictive state used to generate the executing action chunk already contains information about the scene expected at the end of that chunk. This information is available

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) Conventional **(b) Ours**

Figure 1: **From action conditioning alone to recurrent future propagation. (a) Conventional.** A predicted future is used only to generate its accompanying action chunk. Both are conditioned on O_{t_0} and anchored at t_0, but become available only at t_1, leaving part of their horizons stale under inference–execution overlap. **(b) Ours.** FFP propagates the predicted future into a further future that supports both action generation and subsequent propagation.

before the robot reaches that boundary. A useful analogy to human planning is that one need not wait for every intermediate outcome to be fully resolved before reasoning farther ahead. An existing plan can be provisionally extended despite uncertainty, corrected as factual outcomes arrive, and re-anchored by a fresh assessment when continued extension is no longer worthwhile. Crucially, our motivating observations suggest that imperfect predictions can retain task-relevant structure and exhibit cross-boundary continuity at the action-facing interface (Figure 3).

Exploiting this opportunity, however, is non-trivial. A reusable predictive state must advance with control while remaining consistent with physical execution: reusing a prediction unchanged leaves its temporal reference fixed, whereas repeatedly extending imagined futures can accumulate drift (Li et al., 2026). This leads to a central challenge: how to propagate the predictive state into a further future while keeping it grounded in the robot's evolving physical state, thereby reducing the need for repeated native prediction.

We address this challenge with the **Further-Future Propagator (FFP)**, which learns a *future-to-further-future* transition that advances the frozen WAM's action-facing predictive state for the next control round (Figure 1(b)). The updated predictive state supports both the next action chunk and subsequent propagation. FFP advances the predictive state through a residual update grounded in current and historical visual observations and the executing action chunk. Action relevance guides information aggregation from the predictive state, while Stage provides manipulation context to modulate the transformation of aggregated features. Before propagation, Confidence uses the carried predictive state, compressed visual evidence from current and past observations, and the executing action chunk to assess the marginal value of obtaining a fresh predictive state from the native WAM.

We integrate FFP with three architecturally distinct WAMs—VPP, LingBot-VA, and Faster-WAM (Hu et al., 2025; Li et al., 2026; Zhao et al., 2026)—whose action-facing predictive states span hidden representations, future-video latents, and fused key/value states. For Faster-WAM on RoboTwin 2.0 Easy, FFP achieves $7.40\times$ the effective action-generation throughput of RTC while improving success rate from 76.1% to 92.2%. Across four real-world tasks with LingBot-VA and Faster-WAM, task-averaged completion time over successful episodes decreases by 34.3–39.3% relative to the corresponding asynchronous baselines, while mean success rate is maintained for LingBot-VA and improved for Faster-WAM. Our contributions are as follows:

- **Grounded recurrent future propagation.** We formulate WAM predictive futures as recurrent states and introduce FFP to advance them across asynchronous control rounds using visual observations and the executing action chunk, while keeping the propagated state compatible with each WAM's frozen action pathway.
- **Utility-guided predictive refresh.** We learn to estimate the marginal benefit of obtaining a fresh native prediction over reusing the carried predictive state, enabling selective source refresh before propagation rather than a fixed refresh schedule.
- **Cross-architecture and real-world evaluation.** We evaluate FFP on three architecturally distinct WAMs using CALVIN and RoboTwin 2.0, with additional real-world evaluation on two WAMs across four tasks, demonstrating improved control–compute trade-offs over matched asynchronous baselines.

2 Related Work

Predictive future representations in WAMs. WAMs expose predictive information to control through feature conditioning or joint video–action generation (Li et al., 2025; Ye et al., 2026b). VPP and mimic-video condition action policies on representations from video prediction backbones (Hu et al., 2025; Pai et al., 2025). Motus supports joint video–action generation in a unified latent model (Bi et al., 2025), while LingBot-VA autoregressively models video and actions with persistent context (Li et al., 2026). Cross-chunk predictive-state persistence appears in EvoScene-VLA, which carries a scene prior through joint action–scene denoising and corrects the inherited prior against new observations at the next VLM call (Zhang et al., 2026). These formulations expose action-facing predictive states across interfaces. FFP asks whether such states can be advanced across control boundaries while preserving the frozen native predictive-to-action pathway.

Efficient WAM inference. Prior efficiency work operates at several levels: token sparsification and action-aware pruning reduce per-forward computation (Zhu et al., 2026; Pei et al., 2025), feature caching avoids repeated intermediate computation (Ma et al., 2024; Zou et al., 2025; Nawaz et al., 2026), and distillation compresses iterative diffusion policies into few-step action generators (Prasad et al., 2024; Wang et al., 2025). VLA-Cache reuses selected visual K/V representations across observation frames (Xu et al., 2025), while Faster-WAM constructs a fixed future-aware fused K/V interface from a single video-expert pass and reuses it across action-denoising steps (Zhao et al., 2026). FFP makes the updated predictive state itself recurrent: the composed output of each FFP update becomes the input state for the next update, unless a native refresh is selected. Although FFP also uses teacher supervision, it learns this cross-boundary state transition rather than a shorter denoising sampler for the same conditional generation problem.

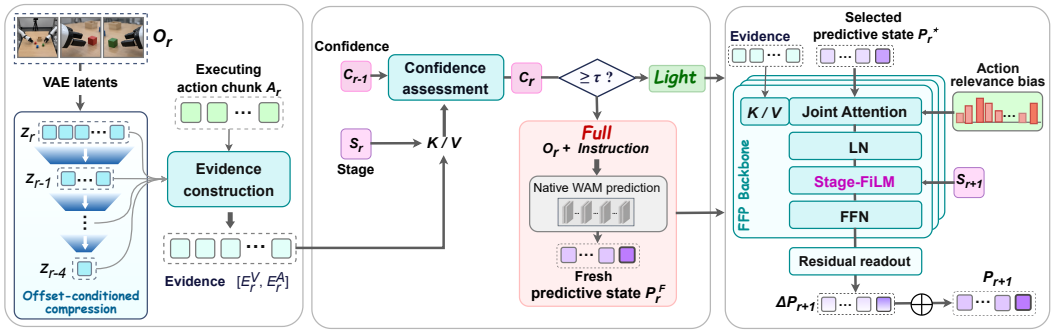
Asynchronous and adaptive robot execution. Real-Time Chunking (RTC) conditions new action chunks on committed actions to maintain continuity during inference–execution overlap (Black et al., 2025). For WAMs, LingBot-VA uses factual observations and the executing action chunk in a forward-dynamics pass to ground the context for subsequent native prediction (Li et al., 2026). Beyond temporal alignment, When to Trust Imagination and TempoWAM adapt execution horizons using future–observation consistency and task progress, respectively (Wang et al., 2026; Ye et al., 2026a). FFP instead acts at the predictive-source level: Confidence estimates the value of obtaining a fresh native prediction before the next state transition. Regardless of the route, the selected source undergoes the same FFP transition before being passed to the frozen native action pathway.

3 Methodology

FFP turns an existing WAM's action-facing predictive state into a recurrent state for subsequent control. While the current action chunk executes, each update performs three operations: *evidence preparation* compresses current and historical factual observations and embeds the executing actions; *source routing* uses Confidence to select the carried state (Light) or a fresh native prediction (Full); and *predictive-state propagation* applies the same grounded residual transition to either source. The composed state conditions the frozen native action pathway and becomes the next recurrent input. We first motivate and formalize this recurrence (Secs. 3.1–3.2), then detail evidence preparation and predictive-state propagation (Sec. 3.3), utility-guided source routing (Sec. 3.4), and carrier-specific instantiation and training (Sec. 3.5). Figure 2 depicts the resulting evidence–routing–propagation pipeline.

3.1 Motivations

Imperfect predictions can retain control-relevant content. Despite blur and visual distortions, the VPP and LingBot-VA predictions capture the approximate locations and coarse shapes of task-relevant objects observed at the end of the corresponding action chunk (Figure 3(a)). Native action attention concentrates near interaction regions (b), while action-facing predictive representations retain substantial similarity across control boundaries (c). Together, these observations motivate retaining predictive content as a prior for cross-boundary propagation.



(a) Evidence Preparation **(b) Source Routing** **(c) Predictive-State Propagation**

Figure 2: **Overview of FFP.** While actions execute, FFP advances predictive states for subsequent control. **(a)** Compressed current and historical observations are combined with executing-action embeddings, providing evidence for routing and propagation. **(b)** Confidence selects the carried state (Light) or a fresh native prediction (Full) based on refresh utility; both enter the same FFP transition. **(c)** Action relevance biases attention over predictive content, while Stage-FiLM modulates post-attention FFN inputs. The predicted residual is added to the selected state, producing a composed state that supports action generation and subsequent propagation. The native WAM remains frozen.

Retention alone is insufficient: static reuse leaves the temporal reference unchanged, while physical execution can depart from the forecast. The static-reuse evaluation in Sec. 4.4 therefore motivates a learned transition. FFP advances the predictive state using factual evidence and control context, while selectively refreshing its source.

3.2 Recurrent Predictive-State Propagation

At control boundary t_r, the robot starts the committed chunk $A_r = [a_{r,1}, \ldots, a_{r,K}]$, where K counts the executed prefix rather than any unused predictions. Inference prepares A_{r+1} for the next boundary t_{r+1} while A_r executes. The carried state P_r conditioned A_r and contains information about the chunk's anticipated outcome before the robot reaches t_{r+1}. FFP advances this predictive prior under factual evidence for subsequent control.

For a frozen WAM (the carrier) indexed by b, let $\mathcal{W}_b$ and Π_b denote its predictive and action pathways. The selected source $P_r^\star$ is either P_r or a fresh native prediction (Sec. 3.4). The *FFP backbone* comprises L_b blocks that update an internal token representation (the workspace), initialized as $H_r^{(0)}$, using visual and action evidence, an action-relevance bias, and Stage-conditioned transformations (Sec. 3.3). A terminal readout maps the final workspace to a residual in native state coordinates:

$$\begin{aligned} \Delta P_{r+1} &= \mathcal{D}_b(H_r^{(L_b)}), \\ P_{r+1} &= P_r^\star + \Delta P_{r+1}, \\ A_{r+1} &= \Pi_b(P_{r+1}; \xi_r^\star). \end{aligned} \tag{1}$$

Here $\xi_r^\star$ contains the instruction and native conditioning context associated with the selected source, such as cached history. The composed state conditions the next action chunk and becomes the next recurrent input. Source addition occurs once after the stack; temporal advancement is learned, not supplied by this identity path. The native WAM initializes the first state and chunk.

3.3 Control-Oriented Predictive-State Transition

Evidence preparation supplies visual and executing-action context for both source routing and predictive-state propagation. Given the selected source, propagation combines relevance-guided aggregation with Stage-conditioned feature evolution.

Evidence preparation. A frozen encoder independently processes each view's latest factual image and up to four historical boundary frames; unavailable entries are masked. Each available frame, captured no later than t_r, uses learned-query compression (Jaegle et al., 2021) conditioned on the number of control steps from capture to the target boundary t_{r+1}. The resulting E_r^V preserves frame

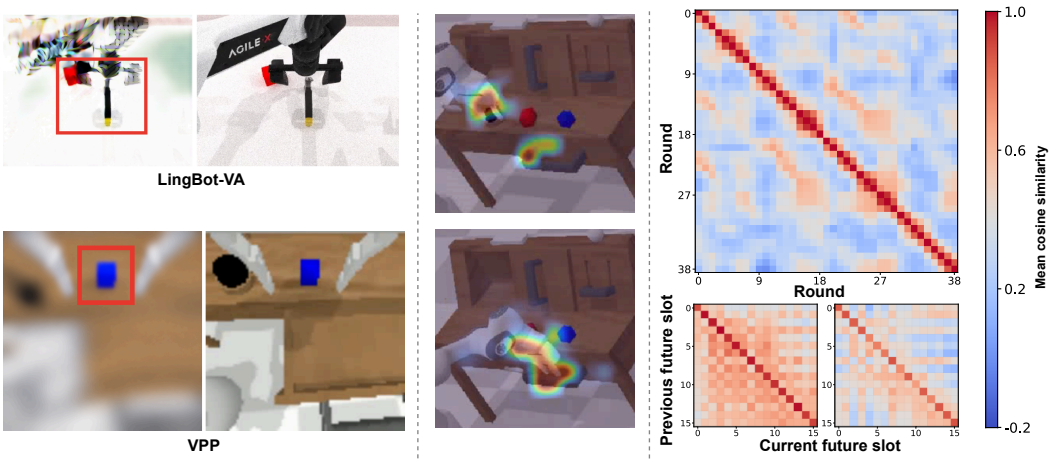


Figure 3: **Motivating observations.** **(a)** Predicted (left) and observed (right) scenes at chunk completion show similar task-relevant object locations and shapes despite prediction blur and distortion. **(b)** VPP action-to-predictive-state attention maps. **(c)** VPP predictive-feature similarity across native refreshes in a successful five-task sequence: a pairwise similarity matrix above and two slot-level comparisons between adjacent refreshes below, sharing a color scale.

and view identity, so subsequent fusion can distinguish evidence by its age relative to the prediction target. Historical encodings are cached; compression is reevaluated for each target. Ordered embeddings of the committed chunk form E_r^A. Both propagation and Confidence use these visual and action features.

Relevance-guided aggregation. The native computation of A_r supplies cached attention R_r over predictive positions. In block ℓ, queries $Q_r^{P,\ell}$ from the evolving workspace read predictive K/V and read-only evidence K/V using scaled dot-product attention (Vaswani et al., 2017):

$$Z_r^{(\ell)} = \mathrm{softmax}\left(\frac{Q_r^{P,\ell}[K_r^{P,\ell}; K_r^{E,\ell}]^\top}{\sqrt{d_k}} + B(R_r)\right)[V_r^{P,\ell}; V_r^{E,\ell}]. \tag{2}$$

The detached, bounded bias $B(R_r)$ applies only to predictive-key logits, favoring action-relevant reads without removing tokens; visual and action evidence remains in the same attention operation. It is a prior from the executing chunk, not exact importance for the next chunk. Predictive K/V are projected from the evolving token state for VPP and LingBot-VA, but from the selected native K/V memory for Faster-WAM; evidence K/V come from $[E_r^V; E_r^A]$.

Stage-conditioned evolution. Approach, fine manipulation, and transport can require different continuations of an interaction. Once per decision, Stage updates its recurrent state from S_r to S_{r+1} using the selected predictive source together with visual and action evidence. This continuous state modulates the normalized input of every block's post-attention FFN through FiLM (Perez et al., 2018). Stage thus conditions feature evolution rather than serving as a K/V source for propagation queries. It is retained across rounds and receives auxiliary next-chunk supervision during training; no discrete stage label is required online.

3.4 Utility-Guided Source Routing

Source routing determines whether native refresh is worthwhile before predictive-state propagation. A large propagation error alone does not establish that refresh would improve the result. Before either candidate update, Confidence reads the carried P_r, incoming Stage S_r, visual evidence, and action evidence to update its recurrent hidden state and produce a propagation score $c_r^{\mathrm{prop}} \in [0, 1]$. A higher score favors continued propagation:

$$P_r^\star = \begin{cases} P_r, & c_r^{\mathrm{prop}} \geq \tau \quad \text{(Light)}, \\ \mathcal{W}_b(\mathcal{I}_r), & c_r^{\mathrm{prop}} < \tau \quad \text{(Full)}. \end{cases} \tag{3}$$

Here $\mathcal{I}_r$ uses only information available at t_r. After source selection, the same FFP transition is applied to either the carried state (Light) or a fresh native prediction obtained from the native WAM (Full). At inference time, only the selected route is executed.

Offline, paired Light/Full actions A^L_{r+1}, A^F_{r+1} are compared with a common teacher A^T_{r+1} obtained at the observed target boundary. Their principal utility is normalized action-MAE reduction:

$$G^A_r = D_A(A^L_{r+1}, A^T_{r+1}) - D_A(A^F_{r+1}, A^T_{r+1}). \tag{4}$$

Together with the analogous predictive-state gain $G^P_r = D_P(P^L_{r+1}, P^T_{r+1}) - D_P(P^F_{r+1}, P^T_{r+1})$, this defines a propagation-acceptability target:

$$q^{\mathrm{acc}}_r = \exp\!\left(-[G^A_r]_+/s_A - \eta_P[G^P_r]_+/s_P\right). \tag{5}$$

Here $[x]_+ = \max(x, 0)$, $s_A, s_P > 0$ scale the gains, and η_P weights state utility. Confidence is trained with binary cross-entropy against this detached target. Larger refresh gains lower propagation acceptability; c^{prop}_r is trained as a routing-utility score. The fresh online source is distinct from the offline target-boundary teacher.

3.5 Instantiation and Training

FFP is instantiated and trained independently for each frozen WAM. The recurrent state is the Video Former's pre-normalization predictive hidden state for VPP, the native future-video latent for LingBot-VA, or eight groups of fused action-facing K/V for Faster-WAM. VPP and LingBot-VA initialize the workspace from the selected predictive state, whereas Faster-WAM resets it to the same learned queries at each update to read its persistent K/V memory.

One-step propagation. During training, we denote the composed state and decoded action from Eq. 1 by $\widehat{P}_{r+1}$ and $\widehat{A}_{r+1}$, and supervise these outputs using target-boundary teachers together with an auxiliary Stage readout:

$$\mathcal{L}_{\mathrm{I}} = \lambda_p D_P(\widehat{P}_{r+1}, P^T_{r+1}) + \lambda_a D_A(\widehat{A}_{r+1}, A^T_{r+1}) + \lambda_s \mathcal{L}_{\mathrm{stage},r+1}. \tag{6}$$

D_P measures MSE in native predictive coordinates normalized by fixed training-set scales; D_A measures normalized action MAE. Action gradients pass through the frozen action pathway to FFP. Future observations and Stage labels supply offline supervision only.

Recurrent training. We unroll the same transition with reduced teacher forcing, including carried-source and post-refresh updates. Composed states recur, with state, action, and Stage losses at supervised boundaries. Factual observations remain paired with the controls actually executed.

Utility learning. We freeze propagation, evidence, Stage, and shared projections, then train only Confidence-specific parameters. Paired updates use independent cache copies, matched action-generation randomness, and identical causal evidence. Carrier-specific implementation, auxiliary supervision, and full training procedures are provided in Appendices A and B.

4 Experiments

We evaluate FFP's control–compute trade-offs across WAM architectures and on real robots, then examine what enables recurrent propagation and when native prediction should be refreshed.

4.1 Experimental Setup

Benchmarks and comparisons. We evaluate VPP on CALVIN ABC→D (Mees et al., 2022) and LingBot-VA and Faster-WAM on RoboTwin 2.0 (Chen et al., 2025). For CALVIN, we report cumulative SR1–SR5 and average completed sequence length (Avg. Len.), averaged over three independently seeded evaluation runs, each comprising 1,000 five-instruction chains. For RoboTwin 2.0, we report task-averaged success over 100 trials per task in each setting (Easy or Hard). All maintable task-performance metrics are from our evaluations under the carrier-specific protocols, with matched task sets and evaluation budgets. The matched asynchronous baselines are RTC for VPP

Table 1: **Closed-loop control and effective action-generation throughput.** E/H denotes Easy and Hard, respectively. Task-performance metrics are measured in our evaluation. Eff. gen. follows Eq. 7; Full rate is the percentage of decision rounds using native source refresh.

(a) VPP / CALVIN ABC→D

Method	SR1 ↑	SR2 ↑	SR3 ↑	SR4 ↑	SR5 ↑	Avg. Len. ↑	Full rate (%) ↓	Eff. gen. (steps/s) ↑
Native Sync	96.2	90.8	86.3	81.8	76.8	4.32	–	111.1
RTC	94.9	89.1	83.5	78.3	72.1	4.18	–	65.2
Always-Full FFP	96.6	91.8	86.1	81.2	77.0	4.33	100	103.1
Adaptive FFP	96.4	90.6	85.9	80.6	75.8	4.29	23.8	205.3

(b) RoboTwin 2.0

Carrier	Method	Easy (%) ↑	Hard (%) ↑	Full rate (%) ↓	Eff. gen. (steps/s) ↑
LingBot-VA	Native Sync	90.4	89.4	–	7.1
	FDM Async	89.3	88.7	–	6.2
	Always-Full FFP	90.8	89.3	100	7.1
	Adaptive FFP	**89.6**	**89.2**	25.8 (E) / 26.2 (H)	10.7 (E) / 10.7 (H)
Faster-WAM	Native Sync	91.3	90.2	–	183.8
	RTC	76.1	73.0	–	36.5
	Always-Full FFP	92.5	90.6	100	182.6
	Adaptive FFP	**92.2**	**90.4**	22.4 (E) / 23.6 (H)	270.6 (E) / 268.6 (H)

and Faster-WAM, and FDM Async for LingBot-VA (Black et al., 2025; Li et al., 2026); Native Sync provides a deployment reference. Always-Full refreshes every round; Adaptive uses the fixed threshold $\tau = 0.5$.

Training data. In simulation, Stage II equally mixes recorded trajectories and rollouts collected under FFP-generated actions. Real-world Stages I–II instead replay the task demonstrations, without additional robot rollouts. Appendix B gives the full training and optimization settings.

Effective action-generation throughput. Benchmark runtime accounting uses RTX 5090 path timings. We measure effective action-generation throughput as usable action steps per second of inference time:

$$R_{\text{gen}} = \frac{K_{\text{eff}}}{T_{\text{update}}}, \qquad K_{\text{eff}}^{\text{RTC}} = \max(0, K_{\text{exec}} - fT_{\text{RTC}}). \tag{7}$$

Here K_{eff} is the number of usable action steps produced per update, and T_{update} is the inference latency per control round in seconds. VPP, LingBot-VA, and Faster-WAM execute $K_{\text{exec}} = 10, 32, 28$ steps at $f = 30, 50, 50$ Hz, respectively; Faster-WAM predicts 32 steps but executes only the first 28. Native, FDM, and FFP count the full executed prefix. RTC discounts the prefix elapsed during its own inference and uses $T_{\text{update}} = T_{\text{RTC}}$. Adaptive latency averages Full/Light costs using observed route fractions; Appendix D details measured module and path timings and mixture-derived averages. Speedups use the matched asynchronous baseline.

4.2 Closed-Loop Performance and Efficiency

Adaptive FFP improves the control–compute trade-off across all three carriers (Table 1). On VPP, it improves Avg. Len. from RTC's 4.18 to 4.29 while achieving $3.15\times$ the effective action-generation throughput (205.3 versus 65.2 steps/s). On LingBot-VA, it preserves control in both settings with $1.73\times$ FDM Async's generation throughput on Easy. On Faster-WAM, Easy/Hard success rises from 76.1/73.0% with RTC to 92.2/90.4%, with $7.40\times$ and $7.35\times$ the effective generation throughput, respectively (270.6 and 268.6 versus 36.5 steps/s). Always-Full achieves better task performance than Adaptive at higher inference latency.

4.3 Real-World Manipulation

We evaluate LingBot-VA and Faster-WAM on four PiPER-X tasks: replacing a block in a bowl (T1), stacking three blocks (T2), standing a bottle upright (T3), and nesting cups (T4).

Table 2: **Real-world manipulation on PiPER-X.** Inference runs on RTX 5090. Each configuration has 50 trials per task. Completion time is averaged over successful episodes. T1–T4 follow Sec. 4.3.

		T1		T2		T3		T4		Average	
Carrier	Method	Succ. (%)↑	Time (s)↓	Succ. (%)↑	Time (s)↓	Succ. (%)↑	Time (s)↓	Succ. (%)↑	Time (s)↓	Succ. (%)↑	Time (s)↓
LingBot-VA	Native Sync	72	238.5	56	217.4	54	115.9	76	134.9	64.5	176.7
	FDM Async	72	187.0	50	179.4	46	98.1	74	127.1	60.5	147.9
	Adaptive FFP	68	122.9	52	117.0	52	65.2	70	54.2	60.5	89.8
Faster-WAM	Native Sync	66	61.3	50	69.5	52	31.6	70	32.1	59.5	48.6
	RTC Async	46	49.7	34	53.1	24	24.9	46	26.8	37.5	38.6
	Adaptive FFP	64	28.6	50	40.5	48	15.5	68	16.9	57.5	25.4

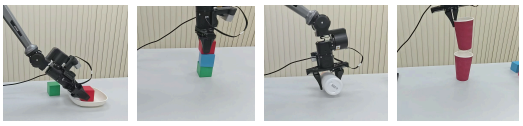


Figure 4: Real-world manipulation tasks.

Each carrier uses task-specific checkpoints adapted with 100 demonstrations per task; FFP uses the same demonstrations without additional robot rollouts. Each configuration receives 50 trials per task, with 32-step chunks at 30 Hz and a 360 s timeout. Table 2 reports success rates over all trials and completion times over successful episodes; Figure 4 shows the task photographs; Appendix E provides the protocol and rollout sequences.

FFP reduces task-averaged successful-episode completion time by 39.3% on LingBot-VA and 34.3% on Faster-WAM relative to FDM Async and RTC, respectively. Average success remains 60.5% on LingBot-VA and increases from 37.5% to 57.5% on Faster-WAM. Completion time decreases on every task for both carriers, while task-level success changes vary. These wall-clock results connect improved inference efficiency to faster successful task completion on physical robots.

4.4 Grounded Recurrent Propagation

Static reuse is insufficient. Alternating native refreshes with one unchanged-state reuse round ($k = 1$), while regenerating actions each round, yields 0.96 Avg. Len. on VPP and one success in 50 LingBot-VA trials. These results motivate updating, rather than merely retaining, the predictive state; Appendix F.1 reports the full reuse-depth study.

Factual evidence drives effective propagation. We retrain five ablated variants at the same backbone depths as the main comparisons, each removing one component (Table 3). Removing visual grounding or executing-action context causes the largest performance drops across all three WAMs, highlighting their importance for grounded recurrent propagation. Among the remaining components, action relevance has the largest effect on VPP, whereas offset conditioning has the largest effect on LingBot-VA and Faster-WAM. Stage provides additional performance gains across all three architectures.

Table 3: **Component ablation of Adaptive FFP.**

Variant	VPP Avg. Len.↑	LingBot-VA Easy (%)↑	Faster-WAM Easy (%)↑
Complete FFP	4.29	89.6	92.2
w/o visual grounding	3.13	67.9	77.7
w/o action context	3.27	79.0	80.4
w/o offset conditioning	4.11	84.3	88.6
w/o Stage	4.04	88.3	91.1
w/o action relevance	3.85	87.2	89.3

Capacity changes both propagation and refresh cost. A larger backbone can lower average inference latency by reducing native refreshes: increasing VPP from one to four blocks improves Avg. Len. from 3.973 to 4.293 while reducing latency from 52.2 to 48.7 ms (Table 4). We use four, four, and two FFP blocks for VPP, LingBot-VA, and Faster-WAM, respectively. Increasing depth beyond these configurations improves neither task performance nor average inference latency. Hard-setting depth sensitivity is reported in Appendix Table 6.

4.5 Utility-Guided Source Refresh

Learned propagation still needs source refresh. Without refresh after initialization, Always-Light reaches 0.96 Avg. Len. on VPP and 10.3% and 34.9% Easy success on LingBot-VA and

Table 4: **FFP-backbone capacity and refresh cost.** All configurations use Adaptive routing with $\tau = 0.5$ and fixed auxiliary topology. RoboTwin 2.0 results are for the Easy setting; Hard results appear in Appendix Table 6. Latency is the average inference latency; VRAM is peak GPU memory usage over the complete run. Bold marks the configurations used in the main comparisons.

(a) VPP / CALVIN

Blocks	Avg. Len.	Full rate (%)	Lat. (ms)	VRAM (GiB)
1	3.973	31.8	52.2	5.561
2	4.096	25.8	49.8	5.568
4	**4.293**	**23.8**	**48.7**	**5.582**
6	4.196	24.0	50.2	5.596

(b) LingBot-VA / RoboTwin

Blocks	Easy (%)	Full rate (%)	Lat. (ms)	VRAM (GiB)
1	83.8	35.1	3171.5	19.38
2	87.1	28.8	3045.8	19.69
4	**89.6**	**25.8**	**2988.2**	**20.30**
6	88.3	25.8	2989.3	20.91

(c) Faster-WAM / RoboTwin

Blocks	Easy (%)	Full rate (%)	Lat. (ms)	VRAM (GiB)
1	89.3	22.9	103.50	22.586
2	**92.2**	**22.4**	**103.47**	**22.604**
4	91.7	24.8	105.26	22.630
6	90.3	22.3	104.25	22.648

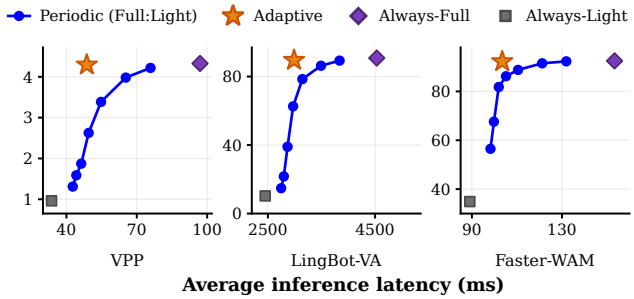


Figure 5: **Adaptive refresh improves the control–compute trade-off.** CALVIN Avg. Len. and RoboTwin 2.0 Easy success (%) versus average inference latency on an RTX 5090. FFP latencies average the measured Full/Light path latencies using their routing fractions. All FFP schedules share the same propagation mechanism.

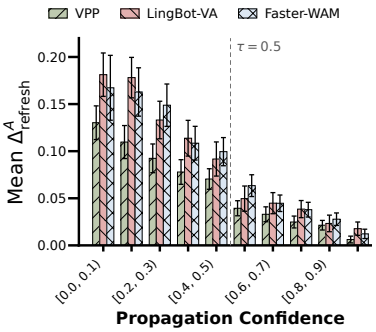


Figure 6: **Held-out Confidence–utility alignment.** Mean teacher-relative action-error reduction from source refresh by Confidence bin, with 95% bootstrap confidence intervals.

Faster-WAM, respectively. We compare Adaptive with periodic Full:Light schedules using the same FFP transition to test whether choosing *when* to refresh improves control.

Figure 5 shows that Adaptive improves task performance at similar refresh budgets and reduces inference latency at comparable performance. On VPP, Adaptive has a Full-refresh rate of 23.8% versus 25.0% for periodic 1:3, yet improves Avg. Len. from 2.63 to 4.29. Relative to periodic 2:1, it reduces average inference latency from 75.87 to 48.70 ms (35.8%) while improving Avg. Len. from 4.22 to 4.29. At similar Full-refresh rates, Adaptive also improves RoboTwin Easy success over periodic 1:3 from 62.6% to 89.6% for LingBot-VA and from 86.2% to 92.2% for Faster-WAM. Supplementary refresh results and Always-Light diagnostics are in Appendix F.

Held-out paired evaluation (Figure 6) further shows that low Confidence identifies decisions with larger teacher-relative action benefit from refresh (Eq. 4; Appendix B.4). This diagnostic explains the score's utility ordering, while the closed-loop comparisons establish the benefit of adaptive refresh allocation.

5 Conclusion

We introduced the Further-Future Propagator (FFP), which treats an existing WAM's action-facing predictive future as a recurrent state. Its *future-to-further-future* transition advances this state using factual visual evidence and executing-action context, while a utility assessment determines when native source refresh is worthwhile. This design reduces repeated native prediction while preserving the frozen WAM's predictive-to-action interface. Across three architecturally distinct WAMs, FFP improves the control–compute trade-off over the corresponding asynchronous baselines. Real-world evaluations on two WAMs also show shorter task-averaged completion times over successful episodes, with mean success maintained or improved. More broadly, these results suggest that predictive futures can serve not only as look-ahead for action generation, but also as persistent computational state that can be grounded and carried forward across control rounds through lighter recurrent updates and selective native refresh.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and polishing, and with literature retrieval and discovery. The authors reviewed the AI-assisted outputs, verified the cited sources, and take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

Implementation and supervision details are provided in Appendix A, with training and counterfactual evaluation procedures in Appendix B. Section 4.1 specifies the evaluation settings and metrics. Appendices D, E, and F provide runtime accounting, real-world protocols, and supplementary propagation and refresh results, respectively.

REFERENCES

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://arxiv.org/abs/1506.03099.

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.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. In *Advances in Neural Information Processing Systems*, volume 38, pp. 33383–33407, 2025. doi: 10.52202/085713-1122. URL https://arxiv.org/abs/2506.07339.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. 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.

Bradley Efron. Nonparametric Standard Errors and Confidence Intervals. *Canadian Journal of Statistics*, 9(2):139–158, 1981. doi: 10.2307/3314608. URL https://onlinelibrary.wiley.com/doi/10.2307/3314608.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video Prediction Policy: A Generalist Robot Policy with Predictive Visual Representations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25g.html.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

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*, 2026. URL https://arxiv.org/abs/2601.21998.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified Video Action Model. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.074. URL https://www.roboticsproceedings.org/rss21/p074.html.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating Diffusion Models for Free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15762–15772, 2024. URL https://arxiv.org/abs/2312.00858.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108. URL https://ieeexplore.ieee.org/document/9788026.

Umair Nawaz, Ahmed Heakl, Ufaq Khan, Abdelrahman Shaker, Salman Khan, and Fahad Shahbaz Khan. WorldCache: Content-Aware Caching for Accelerated Video World Models. *arXiv preprint arXiv:2603.22286*, 2026. URL https://arxiv.org/abs/2603.22286.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-Action Models for Generalizable Robot Control Beyond VLAs. *arXiv preprint arXiv:2512.15692*, 2025. URL https://arxiv.org/abs/2512.15692.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318. PMLR, 2013. URL https://proceedings.mlr.press/v28/pascanu13.html.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware Dynamic Pruning for Efficient Vision-Language-Action Manipulation. *arXiv preprint arXiv:2509.22093*, 2025. URL https://arxiv.org/abs/2509.22093.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11671. URL https://ojs.aaai.org/index.php/AAAI/article/view/11671.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency Policy: Accelerated Visuomotor Policies via Consistency Distillation. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.071. URL https://www.roboticsproceedings.org/rss20/p071.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

Rui Wang, Yue Zhang, Jiehong Lin, Kuncheng Luo, Jianan Wang, Zhongrui Wang, and Xiaojuan Qi. When to Trust Imagination: Adaptive Action Execution for World Action Models. *arXiv preprint arXiv:2605.06222*, 2026. URL https://arxiv.org/abs/2605.06222.

Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-Step Diffusion Policy: Fast Visuomotor Policies via Diffusion Distillation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63399–63416. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25ba.html.

Ronald J. Williams and Jing Peng. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, 2(4):490–501, 1990. doi: 10.1162/neco.1990.2.4.490. URL https://doi.org/10.1162/neco.1990.2.4.490.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient Vision-Language-Action Manipulation via Adaptive Token Caching. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5484. URL https://arxiv.org/abs/2502.02175.

Feng Ye, Yiming Zhao, Yong Yu, Hongxu Zhou, Yong Pan, Yuan Xue, Peng Jia, and Chuanmin Jia. Rethink Before You Execute: Adaptive Execution for World Action Models. *arXiv preprint arXiv:2608.09492*, 2026a. URL https://arxiv.org/abs/2608.09492.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World Action Models are Zero-shot Policies. *arXiv preprint arXiv:2602.15922*, 2026b. URL https://arxiv.org/abs/2602.15922.

Chushan Zhang, Ruihan Lu, Jinguang Tong, Xuesong Li, Yikai Wang, and Hongdong Li. EvoScene-VLA: Evolving Scene Beliefs Inside the Action Decoder for Chunked Robot Control. *arXiv preprint arXiv:2605.21862*, 2026. URL https://arxiv.org/abs/2605.21862.

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*, 2026. URL https://arxiv.org/abs/2608.04404.

Haowei Zhu, Ji Liu, Ziqiong Liu, Dong Li, Jun-Hai Yong, Bin Wang, and Emad Barsoum. Diff-Sparse: Accelerating Diffusion Transformers with Learned Token Sparsity. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2604.03674.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating Diffusion Transformers with Token-wise Feature Caching. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.05317.

A Implementation and Supervision Details

A.1 FFP configuration and factual evidence

FFP is designed and trained separately for VPP, LingBot-VA, and Faster-WAM. The shared formulation does not imply shared weights, identical native token layouts, or a common absolute hidden width. Table 5 records the configurations used in the main benchmark comparisons. Only FFP-backbone depth is varied in the capacity study; compression, Stage, and Confidence retain their single-instance topology.

Table 5: **FFP benchmark configurations.** Parameter counts refer to the complete carrier-specific FFP, not the frozen WAM or action pathway.

Setting	VPP	LingBot-VA	Faster-WAM
FFP blocks	4	4	2
FFP parameters	9.75M	794.34M	55.71M
Latest factual frame slots per view	1	1	1
Historical factual frame slots per view	4	4	4
Propagation threshold τ	0.5	0.5	0.5
Predicted action steps	10	32	32
Executed prefix steps	10	32	28
Controller frequency	30 Hz	50 Hz	50 Hz

Capacity and refresh cost. Propagation capacity affects both Light-path cost and the frequency of native source refresh. Table 4 reports the resulting control–compute trade-offs across backbone depths. For RoboTwin 2.0, each carrier uses the same backbone depth in Easy and Hard, with Hard-setting depth sensitivity reported in Table 6. Appendix D gives the measured path timings and mixture-based runtime accounting.

Factual evidence construction. Each camera view has one slot for its latest factual image and four historical slots for factual frames retained at the immediately preceding control boundaries $t_{r-1}, \ldots, t_{r-4}$. Let $I_{r,j}^{(v)}$ denote an available image from view v, with $j = 0$ for the latest image and $j \in \{1, \ldots, 4\}$ for boundary-indexed history. The recorded capture timestamp $t_{r,j}^{\mathrm{cap},v} \leq t_r$, rather than the boundary index alone, determines the target-relative offset:

$$d_{r,j}^{(v)} = N_{\mathrm{ctrl}}(t_{r,j}^{\mathrm{cap},v}, t_{r+1}), \tag{8}$$

where N_{ctrl} counts control steps from capture to the next target boundary associated with the committed action chunk. A binary validity mask $m_{r,j}^{(v)}$ marks available images; missing entries, including unfilled history at episode start, are masked from propagation, Stage, and Confidence reads. Historical entries contain only factual images available at their respective boundaries. This target-relative conditioning distinguishes when a factual frame was acquired from when its evidence will be used for prediction. The carrier's frozen visual-latent encoder processes each selected image independently:

$$z_{r,j}^{(v)} = \mathcal{E}_{\mathrm{vis}}^{(b)}(I_{r,j}^{(v)}). \tag{9}$$

For each view and capture time, learned queries use the same single-block compression topology,

$$H_{r,j}^{V,(v)} = \mathcal{C}_\phi\Big(z_{r,j}^{(v)}; e(d_{r,j}^{(v)}), e_v\Big), \tag{10}$$

where e_v preserves view identity and $e(d_{r,j}^{(v)})$ modulates normalized compression queries before cross-attention and again in the feed-forward update. Compression queries read K/V only from their own frame's encoder features. Cross-view, cross-time, predictive-state, action, Stage, and Confidence interactions occur in the subsequent FFP attention. Current and historical evidence are organized as

$$\begin{aligned} X_r^V &= [H_{r,0}^{V,(1)}, \ldots, H_{r,0}^{V,(V)}], \\ H_r^V &= [H_{r,j}^{V,(v)}]_{v,\, j>0}, \qquad E_r^V = [X_r^V; H_r^V]. \end{aligned} \tag{11}$$

Table 6: **Hard-setting depth sensitivity on RoboTwin 2.0.** Hard success rate (%) for the same Adaptive configurations evaluated in Table 4. Bold entries mark the carrier-specific configurations used in the main benchmark comparisons.

Blocks	LingBot-VA	Faster-WAM
1	79.3	88.0
2	85.7	**90.4**
4	**89.2**	88.3
6	88.3	90.7

Table 7: **Canonical FFP attention topology.** Visual and action evidence provides read-only K/V. Confidence additionally reads the incoming Stage state; neither auxiliary state is a K/V source for propagation or Stage queries.

Query / K,V	F	X	H	E^A	S	C
Propagation workspace	✓	✓	✓	✓	×	×
Current visual	–	–	–	–	–	–
Historical visual	–	–	–	–	–	–
Action evidence	–	–	–	–	–	–
Stage	✓	✓	✓	✓	×	×
Confidence	✓	✓	✓	✓	✓	×

Historical encoder outputs are cached, so only newly available images require fresh encoding; offset-conditioned compression is reevaluated for the current target boundary before routing. All selected images are factual observations available by t_r.

For VPP, four input VAE tokens correspond to one compression query, giving $M_{\mathrm{VPP}} = N_{\mathrm{VPP}}/4$ when the token count is divisible by four. For the Wan-based LingBot-VA and Faster-WAM instances, query allocation is defined on their native current-image VAE organizations, with carrier-specific input and output projections. Absolute VPP token counts are not transferred to these interfaces. Factual compression queries are distinct from Faster-WAM's reset propagation-workspace queries. With V views and a common per-frame query count M_b, the factual evidence has $5VM_b$ nominal compressed-token slots; tokens belonging to unavailable frame groups are excluded by the validity mask.

The 4:1 VPP compression ratio specifies token count only; it does not imply a fixed non-overlapping 2×2 spatial pooling correspondence. VAE-grid locations indicate approximate receptive-field regions rather than exact image patches.

Ordered executing-action evidence. The committed chunk $A_r = [a_{r,1}, \ldots, a_{r,K}]$ supplies the controls between source and target boundaries. When the carrier predicts a longer chunk, A_r denotes its executed prefix; unused predictions are not intervening controls. We preserve within-chunk order through

$$e^A_{r,k} = E_a(a_{r,k}) + p_k, \qquad E^A_r = [e^A_{r,1}, \ldots, e^A_{r,K}], \tag{12}$$

where E_a uses the carrier's native action embedding when available and p_k is a fixed positional encoding. These are executing or committed controls, not a requirement to wait for the chunk to finish. Their embedded evidence is supplied to both propagation and Confidence.

A.2 Attention topology and auxiliary state updates

Table 7 records the permitted attention edges after visual compression. Rows are query groups and columns are K/V sources; F denotes the selected predictive memory, X current visual evidence, H historical visual evidence, and E^A executing-action evidence. The workspace consists of the internal tokens updated by the FFP backbone. This workspace and the predictive source coincide in the token-state instances, but are distinct for Faster-WAM's structured K/V interface. A checkmark denotes an allowed edge, a cross a prohibited edge, and a dash a passive evidence group with no queries in the main fusion.

The table describes attention K/V exposure; recurrent-state dependence and Stage modulation are separate connections. Incoming S_r and C_{r-1} remain the recurrent states of their respective updates even though they are not exposed as self-K/V. Stage conditions the propagation workspace through the post-attention FFN FiLM of Eq. 18; Confidence is not a feature source for propagation or Stage. In the token-state instances, predictive-source projections are formed from the evolving workspace. Faster-WAM instead projects the selected native K/V memory for reads from reset learned queries. Projections, positional conventions, and native layout operations are included in the block notation.

Cached source-side action relevance. When the native action pathway produces the executing chunk A_r from P_r, we retain post-softmax attention from its final action-generation computation, using the last denoising evaluation for an iterative action head. For VPP and LingBot-VA, $\mathcal{J}_b$ contains all native action-attention layers that read the predictive positions represented by P_r: the Action Head's predictive cross-attention layers for VPP and the action stream's video-reading layers for LingBot-VA. This architectural set is fixed across examples. Within each layer ℓ, we average over all H_ℓ heads and the action queries $\mathcal{Q}_\ell^{\text{exec}}$ corresponding to the committed execution prefix, then average the layers equally:

$$R_{r,j} = \frac{1}{|\mathcal{J}_b|} \sum_{\ell \in \mathcal{J}_b} \frac{1}{H_\ell |\mathcal{Q}_\ell^{\text{exec}}|} \sum_{h=1}^{H_\ell} \sum_{i \in \mathcal{Q}_\ell^{\text{exec}}} a_{r,ij}^{(\ell,h)}, \tag{13}$$

where $a_{r,ij}^{(\ell,h)}$ is the native attention probability for predictive position j. Unused predicted actions do not contribute. LingBot-VA retains only the current predictive chunk's positions, excluding older cache entries. We aggregate the native probabilities before normalization, without separately renormalizing individual heads over the retained predictive subset. For Faster-WAM, averaging over heads and action queries is performed separately at each of the eight native K/V interaction groups; j retains both group and within-group token identity, and different groups are not averaged into a common map. Extraction uses the action forward already required to generate A_r, with extraction and storage included in inference latency.

Let $\mathcal{F}$ contain the valid predictive keys participating in the FFP attention operation and $N = |\mathcal{F}|$. We center relevance relative to its mean and clip the resulting bias:

$$\bar{R}_r = \frac{1}{N} \sum_{j \in \mathcal{F}} R_{r,j}, \qquad g_j(R_r) = \text{clip}\left(\frac{R_{r,j} - \bar{R}_r}{\max(\bar{R}_r, \epsilon_R)}, -1, 1 \right). \tag{14}$$

We compute these statistics in FP32 with $\epsilon_R = 10^{-8}$; visual evidence, action evidence, and masked positions are excluded from the mean. When $\bar{R}_r \geq \epsilon_R$, this equals $\text{clip}(N p_{r,j} - 1, -1, 1)$ for $p_{r,j} = R_{r,j} / \sum_{k \in \mathcal{F}} R_{r,k}$. Uniform relevance, including an all-zero summary, gives $g(R_r) = \mathbf{0}$. The transform is bounded and preserves ordering, allowing ties from clipping. For query i and key j, we add the detached bias before softmax:

$$\ell'_{ij} = \frac{q_i^\top k_j}{\sqrt{d_k}} + \mathbb{I}[j \in \mathcal{F}]\, \alpha\, g_j(\text{sg}(R_r)), \tag{15}$$

with $\alpha = 0.5$, giving an added predictive-key bias in $[-0.5, 0.5]$ and zero added bias on evidence keys. This is $B(R_r)$ in Eq. 2; existing attention masks remain in force. Uniform relevance introduces no bias, while nonuniform relevance may change both the relative predictive reads and their competition with visual and action evidence. All workspace queries keep the same update structure; no token is removed, selectively frozen, or assigned extra computation.

The cached summary reflects the action calculation that produced A_r, not exact importance for the next action chunk. In particular, native index compatibility with a refreshed VPP source does not guarantee identical relevance after its feature content changes. We therefore use a bounded weak prior, not a causal-importance or reliability estimate. It is used after either source choice. The action computation selected for execution supplies R_{r+1} for the next update.

Stage update and readout. A single recurrent Stage query reads the selected source's first-block predictive K/V together with visual and action evidence:

$$S_{r+1} = \mathcal{G}_S(S_r; K_r^{P,1}, V_r^{P,1}, E_r^V, E_r^A). \tag{16}$$

Its auxiliary next-chunk regime readout is

$$p_{r+1}^{\text{stage}} = \text{softmax}(W_s S_{r+1} + b_s). \tag{17}$$

With $Y_r^{(\ell)} = H_r^{(\ell-1)} + W_O^{(\ell)} Z_r^{(\ell)}$, the post-attention update is

$$H_r^{(\ell)} = Y_r^{(\ell)} + \text{FFN}_\ell\Big(\text{FiLM}_{\text{stage}}^{(\ell)}(\text{LN}_\ell(Y_r^{(\ell)}); S_{r+1})\Big). \tag{18}$$

Stage can run alongside first-block propagation attention because it does not require that attention's output. The continuous S_{r+1}, rather than the discretized class, modulates post-attention FFNs throughout the same FFP invocation and is retained for the next update. Stage is evaluated once, independently of backbone depth. It leaves the preceding attention logits unchanged, although its effect on the workspace can influence later attention in a multi-block stack. The state-level source addition in Eq. 1 occurs once after the complete backbone and is distinct from these within-block residual connections.

Assessment order and recurrent bookkeeping. The online evidence cutoff is t_r; t_{r+1} denotes the next nominal target boundary of the committed execution. Target offsets condition the update without exposing the future observation. Confidence is evaluated once before source selection, reading the carried state P_r, incoming Stage S_r, and prepared E_r^V, E_r^A. Its recurrent update and scalar readout are

$$\overline{C}_r = \mathcal{G}_C(C_{r-1}; P_r, S_r, E_r^V, E_r^A), \qquad c_r^{\text{prop}} = \sigma(W_c \overline{C}_r + b_c), \tag{19}$$

where σ is the sigmoid function. The assessment shares the first FFP block's input normalization, positional conventions, and K/V projections. The incoming C_{r-1} is its recurrent state, not a self-K/V token. On Light rounds, the carried-source input K/V are reused for propagation and Stage. On Full rounds, newly computed native source projections replace them for the propagation and Stage updates; Confidence is not reevaluated. This avoids completing a rejected Light backbone and does not introduce intermediate threshold tests or early exits between blocks.

After Light, $C_r = \overline{C}_r$; after Full, it resets to a learned neutral state C_{init}. The scalar score is not fed back as a token. The selected route's updated Stage state is retained in both cases. Visual history, native cache context, the composed predictive state, and relevance summary follow that route. The first predictive state and action chunk come from the original carrier, with auxiliary recurrent states initialized to learned neutral values. Evidence preparation, assessment projections, selected native refresh, all executed FFP blocks, residual readout, and composition are included in inference latency.

Real-world buffering and late-update handling are specified in Appendix E.

A.3 Carrier-specific FFP instantiations

For route $\rho \in \{L, F\}$, let $P_r^L = P_r$ and $P_r^F = \mathcal{W}_b(\mathcal{I}_r)$ denote the carried and fresh sources in Eq. 3. The complete operator includes terminal residual composition and updates the recurrent Stage context:

$$(P_{r+1}^\rho, S_{r+1}^\rho) = \mathcal{S}_\theta(P_r^\rho, E_r^V, E_r^A, S_r; R_r). \tag{20}$$

Each carrier has independent FFP parameters. The workspace $H_r^{(0)}$ is formed from the selected state for VPP and LingBot-VA, but from reset learned queries for Faster-WAM (Table 8). After the FFP backbone, $\mathcal{D}_b$ returns a correction in native predictive coordinates; the composed output supplies both the frozen action interface and the next recurrence. Full supplies a fresh causal source before the same transition, not a target-boundary teacher.

Residual interface coordinates. The terminal readout in Eq. 1 produces a correction with the same shape, index ordering, and scale convention as the selected source. Addition uses the VPP pre-normalization state, the LingBot-VA future-video latent, or the Faster-WAM interval-fused K/V pairs, before the native normalization, cache-writing pass, or direct use of the K/V state by the action pathway, respectively. Faster-WAM retains the key and value conventions separately; its compact memory projections and transient queries are not substituted for the native output coordinates. This index-wise composition does not identify equal slot indices with equal physical timestamps; advancement to the next boundary is learned using factual visual evidence and executing actions, not supplied by the identity path.

Table 8: **Carrier-specific FFP interfaces.** The persistent state is the composed source-plus-correction output, not the residual or transient workspace alone.

Carrier	Recurrent predictive state	Initial workspace	Residual readout	Frozen action interface
VPP	Pre-normalization hidden state	Tokenized selected state	Token-wise LN + linear projection to native hidden coordinates	Original normalization and Action Head
LingBot-VA	Native future-video latent	Tokenized selected state	Token-wise LN + linear projection to native future-video latent coordinates	Terminal cache-writing pass, then action generation
Faster-WAM	Eight fused native K/V groups	Reset learned queries	Group-specific K/V linear projections to native K/V coordinates	Composed K/V directly conditions the action pathway

VPP. For VPP (Hu et al., 2025), we take the Video Former's pre-normalization hidden state as the predictive state. The FFP backbone consists of four randomly initialized blocks following the final Video Former layer's topology; visual compression, auxiliary queries, and evidence fusion are learned for FFP. The residual readout follows Table 8, and the frozen Action Head applies its original normalization to the composed state and supplies the action-relevance summary. On Full, $\mathcal{W}_{\mathrm{VPP}}(O_r;\xi_r)$ provides the fresh source from the latest factual observation O_r, after which the same FFP transition and native action pathway are used.

LingBot-VA. The state retains the native future-video latent organization (Li et al., 2026), and the FFP backbone uses four randomly initialized Wan-compatible FFP blocks. Selected factual images are independently encoded with the frozen Wan VAE and compressed using Eq. 10; this auxiliary evidence path is separate from the carrier's native rolling cache. Light skips iterative native future-video generation. The residual readout follows Table 8; for either route, the composed state retains the layout and temporal positions expected by the action stream. Because that stream reads layer-wise video K/V, the composed state undergoes a terminal native cache-writing pass:

$$\widetilde{\mathcal{K}}^{\rho}_{r+1} = \mathrm{Commit}_{\mathrm{LingBot}}(P^{\rho}_{r+1};\mathcal{K}_r,\xi_r), \tag{21}$$

$$(A^{\rho}_{r+1},\mathcal{K}_{r+1}) = \mathrm{Act}_{\mathrm{LingBot}}(\widetilde{\mathcal{K}}^{\rho}_{r+1};\xi_r). \tag{22}$$

$\mathrm{Commit}_{\mathrm{LingBot}}$ is the frozen terminal Wan cache-writing pass, not an added learned network. It writes K/V for the composed predictive chunk, not the correction alone, under native temporal conventions. Full first obtains

$$P^{F}_{r} = \mathcal{W}_{\mathrm{LingBot}}(\mathcal{I}_r), \tag{23}$$

then uses the same FFP transition, cache-writing pass, and action generation. The FDM-grounded baseline instead reconstructs post-action visual context before native prediction. FDM is a competing asynchronous strategy and is not included in FFP Full.

The rolling cache remains the carrier's single native cache state. FFP-derived entries retain predicted provenance and may condition later processing until factual feedback replaces or invalidates them under the native cache rules. Compact evidence E^{V}_{r} grounds the transition, whereas $\mathcal{K}_r$ retains the longer causal context.

Faster-WAM. The native one-pass predictive pathway exposes interval-fused K/V at eight sparse interaction stages of its 30-layer action pathway (Zhao et al., 2026). These eight action-facing pairs form the persistent predictive state, while the computational workspace is reset at every call:

$$P^{\star}_{r} = \{(K^{\star}_{r,m},V^{\star}_{r,m})\}_{m=1}^{8}, \qquad H^{(0)}_{r} = Q_{\mathrm{learn}}. \tag{24}$$

The same learned query templates are reused, not randomly resampled or carried across calls. Two shared FFP blocks read projected native K/V together with factual visual and executing-action evidence. Source relevance biases these reads and the once-updated Stage state modulates their post-attention FFNs. After the shared FFP backbone, group-specific K/V linear projections produce residual corrections in the corresponding native K/V coordinates:

$$\begin{aligned}(\Delta K_{r+1,m},\Delta V_{r+1,m}) &= \mathcal{D}_m(H^{(L_b)}_{r}),\\ (K_{r+1,m},V_{r+1,m}) &= (K^{\star}_{r,m},V^{\star}_{r,m}) + (\Delta K_{r+1,m},\Delta V_{r+1,m}).\end{aligned} \tag{25}$$

Group-specific readouts operate on the final shared workspace, so backbone depth is independent of the eight native interface groups. The composed native pairs directly condition the frozen action pathway and form the next recurrent state. They are already interval-fused, so Light adds neither a second fusion nor a terminal video-to-cache recomputation. Full obtains fresh fused pairs before applying the same query-based transition. Native token ordering, key/value scales, and positional conventions are preserved. The transient workspace and correction alone are never substituted for the persistent state; Stage, Confidence, and cached relevance retain their auxiliary roles. This interface supplies predictive conditioning memory rather than a reconstructed clean video.

Online update. Algorithm 1 summarizes one update while the committed chunk A_r executes. The native carrier initializes the first predictive state and action chunk, with learned neutral auxiliary states (Appendix A.2). Here ξ includes carrier-native cache and provenance where applicable. $\mathrm{NativeRefresh}_b$ wraps $\mathcal{W}_b$ to return a fresh source and matching context; $\mathrm{NativeAction}_b$ wraps Π_b to retain the updated context and committed-prefix relevance from the same action computation. Workspace initialization and the native action interface follow Table 8.

Algorithm 1 Online FFP Update

Require: Frozen carrier b and trained FFP; propagation threshold $\tau = 0.5$
Require: Carried $(P_r, S_r, C_{r-1}, R_r, \xi_r)$; executing chunk A_r; causal input $\mathcal{I}_r$
Require: Factual image history with capture times $\le t_r$; target boundary t_{r+1}
Ensure: Next chunk A_{r+1} and recurrent $(P_{r+1}, S_{r+1}, C_r, R_{r+1}, \xi_{r+1})$
Prepare E_r^V, E_r^A using Appendix A.1, reusing historical encoder outputs.
Compute $(\overline{C}_r, c_r^{\mathrm{prop}})$ from $(C_{r-1}, P_r, S_r, E_r^V, E_r^A)$ using Eq. 19.
if $c_r^{\mathrm{prop}} \ge \tau$ **then**
    $(P_r^\star, \xi_r^\star, C_r) \leftarrow (P_r, \xi_r, \overline{C}_r)$
else
    $(P_r^\star, \xi_r^\star) \leftarrow \mathrm{NativeRefresh}_b(\mathcal{I}_r; \xi_r)$
    $C_r \leftarrow C_{\mathrm{init}}$
end if
$H_r^{(0)} \leftarrow \mathrm{InitializeWorkspace}_b(P_r^\star)$
Form selected-source first-block predictive K/V; reuse assessment projections on Light.
Update S_{r+1} once using Eq. 16. ▷ May overlap block-1 attention
for $\ell = 1, \dots, L_b$ **do**
    Compute $Z_r^{(\ell)}$ by Eq. 2, using cached R_r as the relevance prior.
    Update $H_r^{(\ell)}$ by the post-attention FFN in Eq. 18, conditioned on S_{r+1}.
end for
$\Delta P_{r+1} \leftarrow \mathcal{D}_b(H_r^{(L_b)})$
$P_{r+1} \leftarrow P_r^\star + \Delta P_{r+1}$
$(A_{r+1}, R_{r+1}, \xi_{r+1}) \leftarrow \mathrm{NativeAction}_b(P_{r+1}; \xi_r^\star)$
return $A_{r+1}, P_{r+1}, S_{r+1}, C_r, R_{r+1}, \xi_{r+1}$

A.4 Automatic Stage-label construction

Stage supervision describes a coarse manipulation regime rather than successful contact or task completion. Its role is to organize the continuous state S_{r+1} used by Stage-FiLM. The labeling protocol uses inexpensive geometric and kinematic signals, without requiring an oracle for contact or grasp status, mandatory object-motion tracking, or a multimodal language-model annotator.

Signals and temporal scope. For a task-relevant interaction region, let ℓ_t be its distance to the end effector, $\mathbf{p}_t^{ee}$ the measured end-effector position, and g_t the measured gripper aperture or discrete state (open or closed). The notation ℓ_t distinguishes this physical distance from the control-step offset $d_{r,j}^{(v)}$ in the visual compressor. Over a short window of duration W, define

$$\Delta\ell_t = \ell_t - \ell_{t-W}, \tag{26}$$

$$\overline{v}_t = \frac{1}{W} \sum_{i \in \mathcal{W}_t} \|\mathbf{p}_i^{ee} - \mathbf{p}_{i-1}^{ee}\|_2, \tag{27}$$

$$b_t^g = \mathbb{I}\left[\max_{i \in \mathcal{W}_t} g_i - \min_{i \in \mathcal{W}_t} g_i > \epsilon_g\right], \tag{28}$$

Table 9: **Automatic kinematic Stage-label rules.** Rules use temporal context rather than a weighted sum of heterogeneous signals.

Regime	Assignment and transition cues
Approach	The target is outside the interaction neighborhood and distance decreases over the window. Far but stationary or retreating motion does not automatically establish approach.
Fine manipulation	The end effector enters the neighborhood and exhibits bounded local pose adjustment or a gripper transition. A gripper event within the neighborhood takes priority over a sustained-motion assignment.
Transport	After a fine-interaction state, end-effector motion is sustained while the target remains nearby and the gripper state is stable. This denotes a post-interaction motion regime, not a verified grasp or successful object transport.

where $\mathcal{W}_t$ contains the samples in $[t - W, t]$. For discrete gripper states, b_t^g indicates a transition between open and closed states. End-effector orientation changes can identify local pose adjustment without converting angular and translational motion into a tuned weighted score.

In simulation, target-region and robot poses come directly from simulator state. No visual fiducial is needed. For real-robot data collection, robot proprioception supplies end-effector and gripper signals, while calibrated RGB-D or fiducial localization can provide the target-region position. These are offline annotation measurements, not additional online FFP inputs. Measurements are time-aligned before distances or motion features are computed; unreliable localization is excluded from supervision rather than interpreted as a stage change.

Temporal assignment rule. A deterministic labeler maintains a discrete annotation state z_t^{ann} independently of the learned recurrent Stage state. It uses an interaction neighborhood with hysteresis, $\ell_{\mathrm{in}} < \ell_{\mathrm{out}}$: entry requires $\ell_t \leq \ell_{\mathrm{in}}$, exit requires $\ell_t \geq \ell_{\mathrm{out}}$, and the prior neighborhood status is retained between those thresholds. The three regimes are assigned using the rules in Table 9.

The state machine permits repeated transitions: sustained motion followed by local adjustment or release returns to fine manipulation, and a change of task-relevant target starts a new interaction sequence. A closed gripper is not required for transport, so the same coarse semantics can cover pushing and articulated manipulation. Target identity follows the training trajectory's task context; it is not chosen from the model's predicted Stage. Multi-target or multi-arm intervals without a unique regime, contradictory cues, and measurements close to decision thresholds are masked. Reliable target motion can be used as a consistency check but is not a mandatory fourth input.

Table 10 gives the temporal, distance, and motion settings. Distances are measured to the task-relevant interaction region, and continuous gripper aperture is normalized to its calibrated range $[0, 1]$. The transport speed and persistence tests apply together with the preceding fine-interaction, proximity, and stable-gripper conditions in Table 9.

Table 10: **Stage-label configuration.** Durations use physical time rather than a shared frame count across controller frequencies.

Setting	Value
Motion window W (s)	0.20
Interaction entry / exit distance (m)	0.04 / 0.06
Minimum approach distance decrease over W (m)	0.005
Minimum transport speed (m/s)	0.04
Transport persistence (s)	0.10
Continuous-gripper event threshold ϵ_g	0.10
Next-chunk supervision prefix (s)	$\min(0.20, \text{chunk duration})$

Alignment with the learned Stage. The supervised label y_{r+1}^{stage} concerns the next chunk, not the already executing A_r. It is obtained from a short recorded prefix beginning at t_{r+1}; a consistent regime in that local prefix supplies the label, while a mixed transition is masked. Its duration is

bounded by the next chunk and does not use episode success or a distant future outcome. The auxiliary loss is

$$\mathcal{L}_{\text{stage},r+1} = m_{r+1}^{\text{stage}} \, \text{CE}(p_{r+1}^{\text{stage}}, y_{r+1}^{\text{stage}}), \tag{29}$$

where m_{r+1}^{stage} excludes uncertain transitions. At deployment, the continuous S_{r+1} is inferred from the ordinary factual, predictive, and executing-action evidence in Table 7; the discrete labeling state, class label, and physical measurement stream are not supplied to FFP.

This construction is weak kinematic supervision. Proximity and stable gripper motion alone do not prove contact, and fast local motion need not establish transport. Masking uncertain intervals is therefore part of the protocol rather than an attempt to force every frame into a class. The distance, motion, event, persistence, and ambiguity thresholds belong to the labeler configuration, separate from the fixed online Confidence threshold τ.

A.5 Continuity of the VPP predictive interface

Figure 3(c) examines native VPP predictive features in one successful five-task sequence. Its interface has 16 future slots with 14 query tokens per slot, yielding 224 index-aligned tokens. The large upper matrix compares refreshes by averaging cosine similarity over these matched tokens. Across the 38 adjacent-refresh pairs, mean aligned-token similarity is 0.777 and the maximum is 0.954. The two lower matrices, arranged side by side, resolve a high-similarity pair and a moderately lower pair into slot-pair comparisons, averaging over the 14 matching query indices and sharing the color scale. Alignment follows native slot and query indices, not necessarily identical physical timestamps. This single-trajectory diagnostic illustrates persistent but non-uniform structure and motivates retaining predictive content as a prior. It does not establish a multi-seed statistical result, universal token correspondence, or that every residual is small, low-rank, or easy to predict. The complementary Faster-WAM diagnostic in Appendix A.7 probes a different carrier and task under its stated conditions.

A.6 Additional action-to-future attention diagnostics

The main text uses VPP as the representative action-path diagnostic because its action-facing predictive interface admits a direct spatial visualization. We additionally inspect the native LingBot-VA action stream, aggregating action-to-future attention over the predictive positions associated with the future-video chunk and decoding the resulting relevance back to image space. Figure 7 shows two representative examples. As in the VPP diagnostic, the response is spatially non-uniform and elevated around the end effector, manipulated object, and local interaction region rather than uniformly distributed across the predicted scene. This provides additional qualitative support for using native action attention as a weak source-side relevance prior. It does not imply that attention is an exact object mask, a calibrated reliability estimate, or a causal measure of token importance.

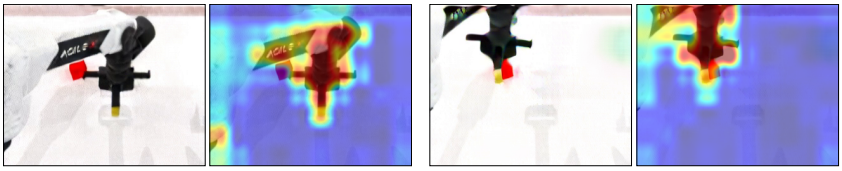


Figure 7: **LingBot-VA action-to-future attention.** Two representative examples show the decoded future-latent relevance used by the native action pathway, with the original view paired with its attention overlay. Responses concentrate around manipulation-relevant regions, complementing the VPP example in Fig. 3(b). The visualization is qualitative and motivates a soft relevance prior rather than hard spatial selection.

A.7 Cross-boundary similarity of Faster-WAM native K/V memory

Faster-WAM exposes a structured action-facing predictive memory rather than a single latent state: SparseMoT reads interval-fused video K/V pairs at a fixed subset of action stages, and the same fused interface is cached and reused during action denoising (Zhao et al., 2026). To inspect how this

native interface evolves across control boundaries, Fig. 8 reports pairwise round-to-round similarity during closed-loop execution of the RoboTwin `open_microwave` task under the Hard setting. For each displayed group and for K/V separately, each heatmap entry is the cosine similarity between two control rounds after matching native future-latent positions, averaged over those matched positions. Thus the diagonal is self-similarity, while off-diagonal structure measures how the same native interface changes as execution advances.

Across groups, the heatmaps show a pronounced diagonal and elevated near-diagonal bands, indicating that adjacent or nearby control rounds retain substantial action-facing structure. Broader off-diagonal similarity generally decreases in later groups, with group-dependent behavior across K and V. This combination supports carrying native K/V across the next boundary while learning a grounded correction, and motivates preserving group-specific readout coordinates rather than collapsing the memory into a single state. We treat the depth-dependent pattern as a diagnostic observation rather than a causal attribution of K/V roles.

We visualize groups 1–7 because group 0 in the released zero-based interaction set corresponds to the singleton first video stage rather than an interval with preceding-depth aggregation. This omission is diagnostic only and does not imply that group 0 lacks conditioning information.

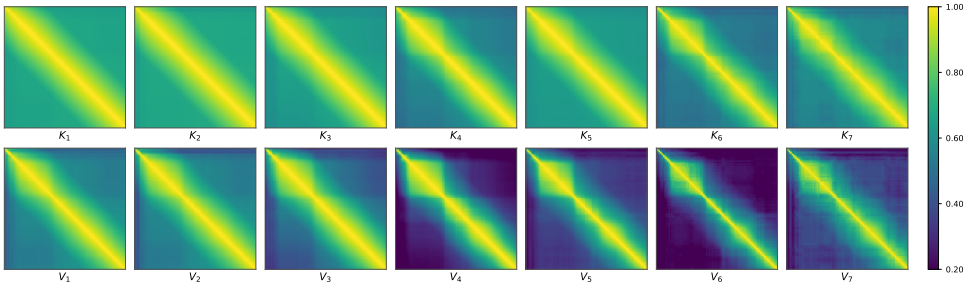


Figure 8: **Cross-boundary similarity of Faster-WAM native K/V memory.** Pairwise round-to-round cosine similarity during closed-loop execution of RoboTwin 2.0 `open_microwave` under the Hard setting. Columns show groups 1–7; the top and bottom rows show K and V. Each entry averages cosine similarity over index-matched future-latent positions. Near-diagonal bands remain high across groups, while broader off-diagonal similarity generally decreases in later groups. Group 0 is omitted because its zero-based interval contains only the first video stage.

A.8 Predictive attention over factual visual evidence

Figure 9 compares carrier-native predictive attention over current factual visual evidence for VPP and LingBot-VA. Despite their different predictive interfaces, both visualizations show spatially non-uniform responses around task-relevant objects, the end effector, and local interaction regions. This signal is distinct from the action-to-future attention used for the online relevance prior R_r: it is reported only as a qualitative diagnostic of where carrier-native predictive pathways attend within the current visual evidence.

No auxiliary attention-guidance loss is used in the checkpoints reported in this paper; Stage is trained only through the supervision and end-to-end gradients described in Appendices A.4 and B.

B Training and Counterfactual Evaluation Details

B.1 Teachers, losses, and optimization scope

In training, $\widehat{P}$ and $\widehat{A}$ denote the composed FFP state and decoded action from Eq. 1; superscript T identifies teacher outputs. The frozen carrier supplies boundary-aligned predictive teachers

$$P_r^T = \mathcal{W}(O_{\le r}; \xi_r^T), \qquad P_{r+1}^T = \mathcal{W}(O_{\le r+1}; \xi_{r+1}^T), \tag{30}$$

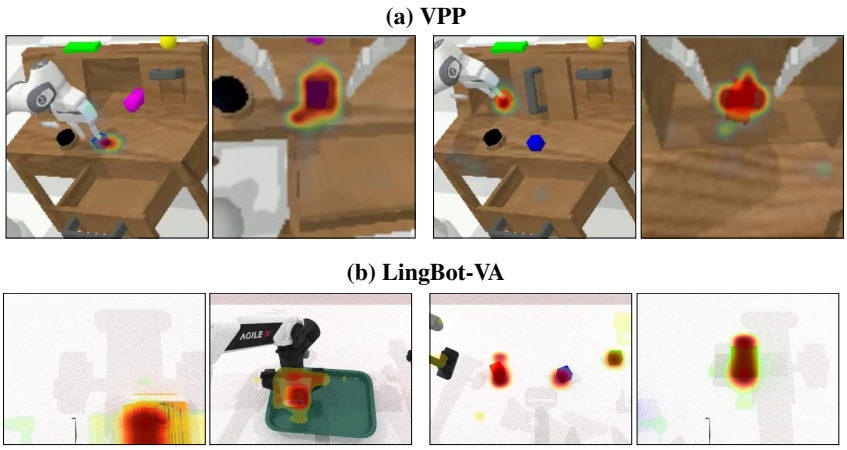


Figure 9: **Predictive attention over current visual evidence.** **(a)** Representative VPP predictive-path attention over factual VAE features. **(b)** Representative LingBot-VA predictive-path attention decoded to image space. The overlays are qualitative diagnostics, not exact object masks, causal attributions, or evidence of an attention-guidance gain.

and the corresponding native action chunks. The teacher at t_{r+1} may use the observation actually reached after executing A_r, whereas the round-r Full candidate may not. Interface layout, predictive horizon, action timestamps, and carrier-native conditioning are matched before comparing outputs. In particular, native source refresh is not a target-boundary teacher and need not have zero teacher-relative error. Residual parameterization changes the predictor's output, not the teacher interface: for source P_r^ρ, the effective correction target is $P_{r+1}^T - P_r^\rho$. We evaluate D_P on $P_r^\rho + \Delta P_{r+1}^\rho$, so the same boundary teacher supervises carried- and fresh-source updates without requiring fixed residual labels.

Action discrepancy is computed in the action normalization already associated with each carrier. Let K_A denote the matched action horizon used for comparison between teacher and student outputs; it is distinct from the executed-prefix count K_{exec} in throughput accounting:

$$D_A(A, A^T) = \frac{1}{K_A D} \sum_{k=1}^{K_A} \sum_{j=1}^{D} |a_{k,j} - a_{k,j}^T|. \tag{31}$$

The same D_A defines action supervision and the held-out action-utility diagnostic.

Fixed-scale native-coordinate state loss. For prediction and teacher tensors X, Y in a matching native layout, define

$$\mathcal{E}_\nu(X, Y) = \frac{1}{|X|} \left\| (X - Y) \oslash \nu \right\|_F^2, \tag{32}$$

where $|X|$ counts scalar entries and $\oslash$ denotes element-wise division with the appropriate broadcasting. We estimate each scale once as the root-mean-square magnitude over all examples and scalar entries of the corresponding target-boundary teacher tensors in the Stage-I training split, with a numerical floor of 10^{-6}, and keep it fixed throughout Stages I–III. VPP and LingBot-VA each use one scalar for their entire native predictive interface; no token- or channel-wise normalization is introduced. The scale is applied identically to the composed prediction and its teacher. For these two carriers, $D_P(P, P^T) = \mathcal{E}_{\nu_b}(P, P^T)$, evaluated in the pre-normalization predictive-hidden and native future-video latent coordinates, respectively. All native tensor entries contribute to the mean.

For Faster-WAM, let $P = \{(K_m, V_m)\}_{m=1}^8$ and $P^T = \{(K_m^T, V_m^T)\}_{m=1}^8$. We compute K and V discrepancies separately and average them with equal weights within and across the eight groups:

$$D_P(P, P^T) = \frac{1}{16} \sum_{m=1}^{8} \left[\mathcal{E}_{\nu_{m,K}}(K_m, K_m^T) + \mathcal{E}_{\nu_{m,V}}(V_m, V_m^T) \right]. \tag{33}$$

Here $\nu_{m,K}$ and $\nu_{m,V}$ are estimated independently by the same RMS rule, yielding 16 fixed scalars across the eight groups. Normalization is confined to the discrepancy calculation: residual composition and native action interfaces retain the original coordinates and scale conventions of the action

and cache interfaces. The same D_P is used in one-step supervision, recurrent supervision, and the predictive-state refresh gain. It is separate from the diagnostic cosine similarity in Fig. 3.

We use $(\lambda_p, \lambda_a, \lambda_s) = (1, 1, 0.05)$ in both Stage I and Stage II, including every supervised boundary in Eq. 35. These coefficients weight predictive-state agreement, normalized action MAE, and auxiliary Stage supervision, respectively. The state-utility weight η_P in Eq. 5 is a separate coefficient.

The original WAM and action parameters have gradients disabled. This does not mean that every operation through those pathways is executed under a global no-gradient context: action consistency must retain derivatives with respect to the propagated input. In LingBot-VA, the terminal cache-writing and action operations therefore transmit gradients through the composed state and source addition to the residual predictor while their parameters remain frozen. For Faster-WAM, the boundary teacher supplies native interval-fused K/V with matching layout and sampling conventions, and the action path transmits gradients with respect to the composed cache. The transient queries have no teacher hidden-state target. Stages I and II update the propagation, evidence, and Stage parameters; Stage III freezes these and the shared projections, updating only Confidence-specific state-update, readout, and initialization parameters.

Optimization settings. We use AdamW (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and weight decay 0.01, excluding biases and normalization parameters from decay. The global gradient norm is clipped (Pascanu et al., 2013) at 1.0. In each stage, the learning rate warms up linearly over the first 5% of optimizer updates, then follows cosine decay (Loshchilov & Hutter, 2017) to 10% of its peak. Table 11 lists peak rates and effective batches, accumulated across devices and microbatches. We use a fixed training seed of 101 for all carrier-specific FFP models. The original carrier remains frozen. Stage III updates only Confidence-specific parameters, with decision samples retaining their ordered recurrent context.

Table 11: **FFP optimization settings.** Peak learning rates and effective batch sizes for the three training stages.

Stage	Peak LR	Effective batch
I	10^{-4}	64 one-step tuples
II	5×10^{-5}	16 recurrent segments
III	10^{-4}	128 decisions

B.2 Recurrent segments and teacher forcing

Stage I trains one-step propagation on offline trajectory tuples; Stage II trains recurrent segments that expose FFP to its own composed predictive states.

Simulation data. In simulation, Stage II incorporates trajectories collected by executing actions decoded from FFP states through the frozen native action pathway and recording their outcomes. Each simulation minibatch mixes recorded-replay and student-rollout segments in a 1:1 ratio. Student rollouts are refreshed every 1,000 optimizer updates using the current propagation parameters and retained in a bounded replay buffer. Recorded segments replayed in Stage II retain the controls that produced their observations, rather than unexecuted predicted actions. Teachers use the corresponding observed boundaries.

Real-world demonstration replay. For the real-world experiments, Stage I and Stage II construct training samples from the same 100 demonstrations per task (Appendix E). Stage II recurrently unrolls the FFP predictive state over contiguous demonstration segments, while factual observations and executing-action evidence remain the paired original action–observation sequence from that demonstration. Student-state recurrence does not substitute unexecuted student actions for the controls that produced the recorded observations. No additional student-policy robot rollouts or perturbed training data are used in either stage.

Recurrent unroll and gradient truncation. We set $K_{\text{train}} = 6$ and sample the segment length $n \sim \mathcal{U}\{1, \ldots, 6\}$ uniformly over the six integer lengths. A segment contains n successive chunk-

level propagation transitions,

$$P_r \longrightarrow \widehat{P}_{r+1} \longrightarrow \cdots \longrightarrow \widehat{P}_{r+n}. \tag{34}$$

This temporal unroll is independent of FFP-backbone depth: a single transition executes four FFP blocks in VPP, four in LingBot-VA, or two in Faster-WAM. For Faster-WAM, each transition resets the learned query workspace, while the composed K/V state and the auxiliary recurrent context follow the selected trajectory. The recurrent objective is

$$\mathcal{L}_{\text{rec}} = \frac{1}{n}\sum_{k=1}^{n}\Big[\lambda_p D_P(\widehat{P}_{r+k}, P^T_{r+k}) + \lambda_a D_A(\widehat{A}_{r+k}, A^T_{r+k}) + \lambda_s \mathcal{L}_{\text{stage},r+k}\Big]. \tag{35}$$

During Stage II, we use scheduled sampling (Bengio et al., 2015): the teacher-forcing probability decreases linearly from 0.5 to 0 over the first 60% of optimizer updates and remains zero for the final 40%. At each recurrent predictive-input boundary, this probability selects the boundary-aligned teacher state; otherwise, the preceding composed student state is carried forward. Switching between teacher and student states keeps predictive state, action conditioning, relevance summary, and native cache provenance coherent. The substituted state is used as the propagation input and added to the predicted residual; carrying only a correction would not implement the recurrence. Teacher forcing does not replace factual observations or recorded executing-action evidence with a continuation from different controls, and its decay does not remove the fresh-source training examples.

We use truncated backpropagation through time (Williams & Peng, 1990) with a fixed gradient span $K_{\text{grad}} = 2$ throughout Stage II. Recurrent tensors are carried forward as values but detached from the preceding computation graph after every two transitions; a shorter final span ends with the segment. Thus forward exposure can reach six transitions while each differentiable span covers at most two. Truncation neither resets the carried predictive values nor forces a native refresh. The forward training horizon does not impose an online Light-streak cap: neither $K_{\text{train}} = 6$ nor the periodic $1{:}6$ evaluation forces Full at the next deployment round.

B.3 Confidence targets with a fixed threshold

At each training decision, the carried-source alternative is

$$(P^L_{r+1}, S^L_{r+1}) = \mathcal{S}_\theta(P_r, E^V_r, E^A_r, S_r; R_r), \quad A^L_{r+1} = \Pi_b(P^L_{r+1}; \xi^L_r), \tag{36}$$

while the fresh-source alternative obtains $P^F_r = \mathcal{W}_b(\mathcal{I}_r)$ and then uses

$$(P^F_{r+1}, S^F_{r+1}) = \mathcal{S}_\theta(P^F_r, E^V_r, E^A_r, S_r; R_r), \quad A^F_{r+1} = \Pi_b(P^F_{r+1}; \xi^F_r). \tag{37}$$

The branches start from independent copies of the same pre-routing context. Each $\mathcal{S}_\theta$ call includes addition to its own selected source: P_r for Light and P^F_r for Full. Utility compares the composed states and their decoded actions, not the magnitudes of the two corrections. Replaying one branch must not mutate the other branch's cache or the executed trajectory. Matched action-generation randomness removes an avoidable source of within-pair discrepancy.

The signed action gain is defined in Eq. 4. We also compute the teacher-relative predictive-state gain,

$$G^P_r = D_P(P^L_{r+1}, P^T_{r+1}) - D_P(P^F_{r+1}, P^T_{r+1}). \tag{38}$$

Positive gain means improved agreement with the common teacher, not a physically perfect forecast. Action improvement is the main signal, with positive predictive-state gain supplying auxiliary supervision. Confidence is trained against the propagation-acceptability target in Eq. 5 using

$$\mathcal{L}_{\text{conf}} = \text{BCE}(c^{\text{prop}}_r, \text{sg}(q^{\text{acc}}_r)), \tag{39}$$

where sg stops gradients through the paired supervision.

Algorithm 2 summarizes a Stage-III supervision step with the ordered recurrent context retained. Each branch uses its own native context and the complete transition $\mathcal{S}_\theta$, including source addition. The target-boundary record enters only teacher construction; fixed gain scales are estimated from the training split as specified below.

Algorithm 2 Paired Refresh-Utility Supervision for Confidence

Require: Training decision $(P_r, S_r, C_{r-1}, R_r, \xi_r, A_r, \mathcal{I}_r)$ with causal factual history
Require: Recorded target-boundary observation and teacher context; fixed s_A, s_P, η_P
Require: Frozen carrier, propagation, evidence, Stage, and shared projections
Prepare shared E_r^V, E_r^A from the causal record at t_r (Appendix A.1).
Compute c_r^{prop} with the incoming recurrent context using Eq. 19.
Make independent copies ξ_r^L, ξ_r^F of the same pre-routing native context ξ_r.
$P_r^L \leftarrow P_r$; $(P_r^F, \xi_r^F) \leftarrow \text{NativeRefresh}_b(\mathcal{I}_r; \xi_r^F)$
Fix matched action-generation randomness for the two candidate action calls.
for $\rho \in \{L, F\}$ **do**
    $(P_{r+1}^\rho, S_{r+1}^\rho) \leftarrow \mathcal{S}_\theta(P_r^\rho, E_r^V, E_r^A, S_r; R_r)$
    $A_{r+1}^\rho \leftarrow \Pi_b(P_{r+1}^\rho; \xi_r^\rho)$, using the matched action-generation randomness.
end for
Obtain common (P_{r+1}^T, A_{r+1}^T) from the recorded target boundary (Appendix B.1).
Compute G_r^A and G_r^P using Eqs. 4 and 38.
Construct q_r^{acc} using Eq. 5 with fixed s_A, s_P, η_P.
Update only Confidence-specific parameters using the detached-target BCE in Eq. 39.

Training-split gain scales. We fix $\eta_P = 0.1$. For each carrier b, let $\mathcal{D}_{\text{train}}^{(b)}$ index training-split counterfactual pairs evaluated with the propagation parameters fixed for Stage III. We form separate collections of strictly positive action and predictive-state gains,

$$\begin{aligned} \mathcal{G}_{A,b}^+ &= \{G_i^A : i \in \mathcal{D}_{\text{train}}^{(b)},\ G_i^A > 0\}, \\ \mathcal{G}_{P,b}^+ &= \{G_i^P : i \in \mathcal{D}_{\text{train}}^{(b)},\ G_i^P > 0\}, \end{aligned} \tag{40}$$

and set

$$s_A^{(b)} = Q_{0.75}(\mathcal{G}_{A,b}^+), \qquad s_P^{(b)} = Q_{0.75}(\mathcal{G}_{P,b}^+). \tag{41}$$

$Q_{0.75}$ denotes the empirical 75th percentile; repeated gain values retain their sample multiplicities. Zero and negative gains are excluded from these scale estimates, not reassigned a positive value. We compute the scales separately for each carrier and hold them fixed for Confidence training and evaluation, sharing them across that carrier's manipulation regimes. Neither test outcomes nor test Full-refresh rates enter this calculation. These gain scales are distinct from the fixed native-coordinate scales ν used inside D_P.

Larger positive refresh gains lower the target for continued propagation. The carrier-specific scales s_A, s_P and fixed weight η_P are determined from training data and are not tuned to test outcomes or Full-refresh rates; the online threshold remains fixed at $\tau = 0.5$. Held-out utility evaluation uses signed action gain alone, separately from the auxiliary state term used during Confidence training, and c_r^{prop} is not interpreted as a calibrated probability of task success.

B.4 Held-out action-utility diagnostic

The diagnostic is separate from Confidence training. During held-out evaluation, the propagation score is recorded before routing and grouped into ten width-0.1 bins, with the final bin including 1.0. Up to 100 naturally occurring decisions per bin are sampled across trajectories for offline paired replay; sparse bins are not padded. Light and Full use identical causal pre-routing evidence, independent cache copies, and matched action-generation randomness. The target-boundary observation is introduced only after the recorded execution reaches that boundary and is used solely to construct the common teacher, so the online router never observes future information.

The plotted $\Delta_{\text{refresh}}^A$ is the signed action gain G_r^A from Eq. 4; positive values favor refresh and negative values favor Light. We report per-bin means with 95% bootstrap confidence intervals (Efron, 1981) and compare trends only within each carrier because action spaces and normalizations differ. Binning evaluates utility ordering rather than task-success calibration.

Figure 6 shows decreasing means from the lowest to highest Confidence bin: 0.1302 to 0.0062 for VPP, 0.1812 to 0.0177 for LingBot-VA, and 0.1673 to 0.0122 for Faster-WAM. Thus lower propagation Confidence identifies decisions with larger teacher-relative action benefit from source refresh, while positive high-Confidence means remain compatible with routing by marginal refresh value rather than requiring Light to match the teacher exactly.

Table 12: **RoboTwin 2.0 horizon-group success rates (%).** No missing group scores are inferred from All.

Carrier	Method	Setting	H1	H2	H3	All
LingBot-VA	Native Sync	Easy	90.7	90.3	89.1	90.4
	Native Sync	Hard	90.3	88.0	88.2	89.4
	FDM Async	Easy	91.5	86.5	84.3	89.3
	FDM Async	Hard	91.8	85.2	80.1	88.7
	Always-Full FFP	Easy	92.1	89.3	87.4	90.8
	Always-Full FFP	Hard	90.5	88.1	85.9	89.3
	Adaptive FFP	Easy	91.8	86.4	85.5	89.6
	Adaptive FFP	Hard	90.7	87.5	85.2	89.2
Faster-WAM	Native	Easy	92.1	90.5	88.9	91.3
	Native	Hard	91.4	88.9	86.9	90.2
	RTC	Easy	78.3	75.8	64.0	76.1
	RTC	Hard	79.3	64.3	61.3	73.0
	Always-Full FFP	Easy	93.6	90.4	92.1	92.5
	Always-Full FFP	Hard	92.0	88.8	87.4	90.6
	Adaptive FFP	Easy	93.5	91.2	87.6	92.2
	Adaptive FFP	Hard	91.7	89.5	85.4	90.4

C RoboTwin Horizon-Group Results

The RoboTwin main table separates Easy and Hard. Table 12 retains the available H1–H3 breakdowns. We use the 50-task grouping with 30, 15, and 5 tasks in H1, H2, and H3; with equal task weights,

$$\mathrm{All} = 0.6\,\mathrm{H1} + 0.3\,\mathrm{H2} + 0.1\,\mathrm{H3}. \tag{42}$$

All values in the table are based on our evaluation; where All is reconstructed from displayed H-group means, we retain sufficient precision internally and round the main-table value to one decimal place. H groups contain different tasks, so they are not nested conditional-success probabilities.

D Detailed Runtime and Action-Generation Throughput

Table 13 consolidates RTX 5090 inference latency and effective action-generation throughput for the benchmark configurations, using the metric defined in Eq. 7.

Source-refresh mixtures. For a periodic schedule with a Full and b Light updates, the mixture-implied latency is

$$\overline{T}(a{:}b) = \frac{aT_{\mathrm{Full}} + bT_{\mathrm{Light}}}{a+b}. \tag{43}$$

Adaptive FFP uses $\overline{T} = p_F T_{\mathrm{Full}} + (1-p_F)T_{\mathrm{Light}}$, where p_F is the observed Full-refresh fraction. Average inference latencies are calculated from measured Full/Light path timings and observed route frequencies, retaining the supplied Full:Light ratios or full-precision fractions.

VPP and RTC. The native forward and four-block FFP module have measured latencies of 90.03 and 6.94 ms, respectively. The measured complete Full and Light paths take 96.97 and 33.66 ms. RTC has a measured latency, $T_{\mathrm{RTC}} = 105.05$ ms. At 30 Hz, the elapsed prefix is $30 \times 0.10505 = 3.1515$ step-equivalents, leaving $10 - 3.1515 = 6.8485$. Hence

$$R_{\mathrm{gen}}^{\mathrm{VPP+RTC}} = \frac{6.8485}{0.10505} = 65.2\ \mathrm{steps/s}. \tag{44}$$

The native rate is $10/0.09003 = 111.1$ steps/s. The observed Adaptive Full:Light ratio 1:3.21 yields $\overline{T} = 48.6980$ ms and $R_{\mathrm{gen}} = 205.3$ steps/s.

LingBot-VA. The native WAM, FDM module, and FFP module have measured latencies of 4509.97, 649.92, and 21.11 ms, respectively. The measured complete FDM Async, FFP-Full, and FFP-Light paths take 5159.89, 4531.08, and 2450.60 ms. Full consists of native source refresh, FFP advancement, and one native cache-writing pass followed by action generation. Light retains the terminal cache and action costs.

The Easy Full:Light ratio 1:2.87 gives

$$\overline{T}_E = \frac{4531.08 + 2.87 \times 2450.60}{3.87} = 2988.19\,\text{ms}. \tag{45}$$

The Hard Full-refresh fraction is 0.262 (26.2%), giving $\overline{T}_H = 2995.69$ ms. Counting 32 actions yields 10.7 steps/s in both settings, versus 6.2 for FDM Async. The displayed 25.8% Easy Full-refresh rate is rounded from the supplied Full:Light ratio. Both settings use the same measured Full/Light path costs, with different observed route fractions. For the separately timed FDM and FFP modules, the latency ratio is $649.92/21.11 \approx 30.8$.

Faster-WAM and RTC. The native forward takes 152.30 ms. Although the policy generates 32 actions, only the first 28 are committed at 50 Hz, so $R_{\text{gen}}^{\text{native}} = 28/0.1523 = 183.8$ steps/s. RTC takes 323.53 ms, yielding

$$K_{\text{eff}}^{\text{RTC}} = 28 - 50 \times 0.32353 = 11.8235, \tag{46}$$

$$R_{\text{gen}}^{\text{RTC}} = 11.8235/0.32353 = 36.5\ \text{steps/s}. \tag{47}$$

The RTC calculation uses continuous-time step-equivalents as defined in Eq. 7.

The two-block FFP has a measured module latency of 1.54 ms. The measured complete Full and Light paths take 153.36 and 89.07 ms, respectively. The observed Full-refresh fractions of 0.224 and 0.236 for Easy and Hard settings give

$$\overline{T}_E = 0.224(153.36) + 0.776(89.07) = 103.47096\,\text{ms}, \tag{48}$$

$$\overline{T}_H = 0.236(153.36) + 0.764(89.07) = 104.24244\,\text{ms}. \tag{49}$$

Consequently, Adaptive provides 270.6 and 268.6 steps/s on Easy and Hard settings, respectively, and Always-Full provides 182.6. Adaptive rates are derived from the Full/Light mixtures above. The 55,706,625 trainable parameters are reported as 55.71M; peak GPU memory usage over the complete run is 23,146.30 MiB (22.604 GiB).

Table 13: **RTX 5090 runtime accounting.** E/H denotes Easy and Hard, respectively. Shared rows use the same measured path timing in both settings. Average FFP inference latencies use measured route fractions; fractional K_{eff} applies only to RTC.

Carrier	Method	Lat. (ms)	K_{eff}	Eff. gen. (steps/s)
VPP	Native Sync	90.03	10	111.1
	RTC	105.05	6.8485	65.2
	Always-Full FFP	96.97	10	103.1
	Periodic 1:3	49.49	10	202.1
	Adaptive FFP	48.70	10	205.3
LingBot-VA	Native Sync	4509.97	32	7.1
	FDM Async	5159.89	32	6.2
	Always-Full FFP	4531.08	32	7.1
	Periodic 1:3 (E)	2970.72	32	10.8
	Adaptive FFP (E)	2988.19	32	10.7
	Adaptive FFP (H)	2995.69	32	10.7
Faster-WAM	Native	152.30	28	183.8
	RTC	323.53	11.8235	36.5
	Always-Full FFP	153.36	28	182.6
	Periodic 1:3 (E)	105.14	28	266.3
	Adaptive FFP (E)	103.47	28	270.6
	Adaptive FFP (H)	104.24	28	268.6

E Real-World Tasks, Protocol, and Qualitative Rollouts

Setup and adaptation. The PiPER-X setup uses one third-person view and one wrist-camera view. Each carrier uses a task-specific checkpoint trained or fine-tuned with 100 demonstrations per task, rather than a multitask checkpoint. FFP Stages I and II use these same demonstrations,

without additional robot rollouts or perturbed training data, with Stage II unrolling predictive states on recorded segments (Appendix B.2). Both carriers generate 32-step chunks and execute at 30 Hz; these settings differ from the RoboTwin runtime configurations in Appendix D. Early FFP updates are buffered until the executing chunk ends. If an update completes late, the robot holds position until the result is available and then executes the full returned committed prefix without truncation. This waiting is included in wall-clock completion time; subsequent updates retain the actual observation timestamps together with state and cache provenance.

Evaluation and aggregation. Each configuration is tested in 50 trials per task (200 trials per configuration; 1,200 trials across both carriers and their three configurations). The episode limit is 360 s. Success is determined by completion of the task instruction within this limit. Completion time is averaged over successful episodes only; failed trials remain in the success-rate denominator but do not contribute a zero or timeout value to the time mean. The Average columns in Table 2 average the four task-level values with equal weights. Average success thus equals total successes divided by 200, whereas Average time is not pooled over successes. Time reductions are calculated from the unrounded task averages before display rounding.

Timeout-penalized task throughput. We jointly evaluate task completion and execution speed using timeout-penalized task throughput. For each task, we divide the number of successful trials by the total accounted time, using recorded completion times for successful trials and the episode limit of $T_{\max} = 360$ s for unsuccessful trials. With task-level success rate p_j and mean successful-episode completion time $\overline{T}_{\mathrm{succ},j}$ in seconds, we report the equal-weight average across the four tasks:

$$\overline{R}_{\mathrm{pen}} = \frac{60}{4} \sum_{j=1}^{4} \frac{p_j}{p_j \overline{T}_{\mathrm{succ},j} + (1 - p_j) T_{\max}}.$$

The resulting metric is expressed in successful tasks per minute and is computed from the same trials as Table 2. As shown in Table 14, Adaptive FFP improves average timeout-penalized task throughput by 20.3% on LingBot-VA and 124.4% on Faster-WAM relative to FDM Async and RTC Async, respectively.

Table 14: **Timeout-penalized real-world task throughput.** Values average the four task-level throughputs with equal weights; unsuccessful trials are assigned 360 s.

Carrier	Method	Throughput (successful tasks/min) ↑
LingBot-VA	Native Sync	0.165
	FDM Async	0.162
	Adaptive FFP	**0.194**
Faster-WAM	Native Sync	0.217
	RTC Async	0.098
	Adaptive FFP	**0.220**

F Supplementary Propagation and Refresh Results

This section supplements the propagation and refresh studies in Secs. 4.4 and 4.5. Static reuse holds the predictive state unchanged, whereas Always-Light applies the learned FFP transition without native refresh. Periodic schedules retain the same transition and replace Confidence with a fixed refresh schedule.

Stage ablation. The *w/o Stage* variant in Table 3 removes the recurrent Stage state and its auxiliary supervision, disables Stage-conditioned FiLM modulation, and removes the Stage input to Confidence.

F.1 Static predictive-state reuse

After each native-prediction round, we reuse the action-facing predictive state unchanged for k rounds, regenerating actions through the native action pathway at each round.

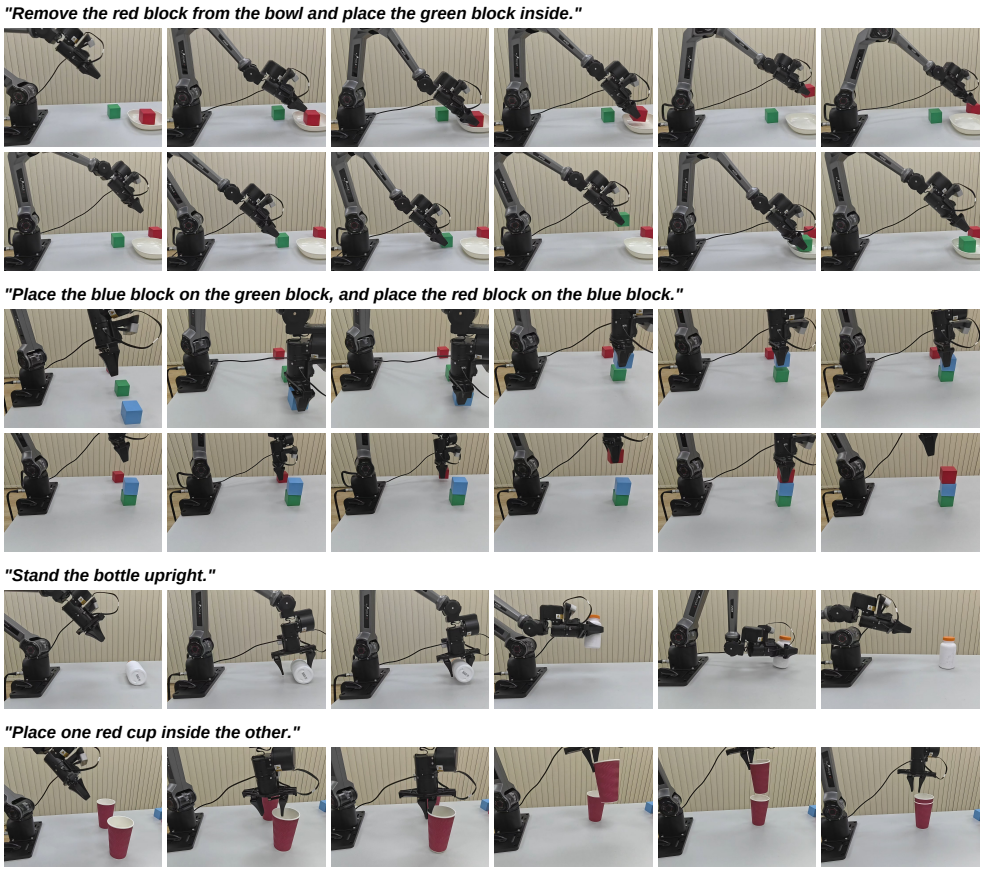


Figure 10: **Qualitative real-world rollouts on PiPER-X.** Sequences follow tasks T1–T4 defined in Sec. 4.3, with full instructions above each sequence. Third-person frames illustrate task progress; frame spacing is not a calibrated time axis. Quantitative results appear in Table 2.

Table 15: **Static predictive-state reuse.** VPP average completed sequence length for k reuse rounds.

Reuse rounds k	1	2	3	4	5	6
VPP Avg. Len.	0.96	0.67	0.38	0.15	0.10	0.06

At $k = 1$, VPP's cumulative SR1–SR5 are 53%, 26%, 14%, 2%, and 1%. LingBot-VA completes one of 50 trials at $k = 1$, with no observed successes for $k = 2, \ldots, 6$.

F.2 Propagation without native refresh

Always-Light runs the learned transition without native source refresh after initialization. On VPP, SR1–SR5 are 56.7%, 24.1%, 11.3%, 3.2%, and 0.9%, giving an unrounded Avg. Len. of 0.962 (0.96 when displayed to two decimals). LingBot-VA reaches 10.3% Easy-All success, with 14.2%, 5.3%, and 1.5% on H1, H2, and H3, respectively. Faster-WAM reaches 34.9% Easy-All.

F.3 Fixed source-refresh schedules

We evaluate periodic Full:Light schedules 2:1 and 1:k, $k \in \{1, \ldots, 6\}$, using the same FFP transition as Adaptive. Figure 5 reports the performance–latency trade-offs on CALVIN and RoboTwin 2.0 Easy. For VPP, periodic 1:3 yields SR1–SR5 of 75.2%, 69.2%, 51.7%, 41.6%, and 24.9%. On LingBot-VA, Adaptive improves Easy success over periodic 1:3, with gains of 22.4, 31.3, and 41.2 percentage points on H1, H2, and H3, respectively.

F.4 Confidence threshold sensitivity

For each carrier, we keep the trained FFP and Confidence checkpoint fixed and vary only the routing threshold τ at test time, without retraining. We use the same tasks, evaluation seeds, and budgets as in the main comparisons. Figure 11 shows that increasing τ assigns more decisions to Full refresh, changing the control–compute operating point.

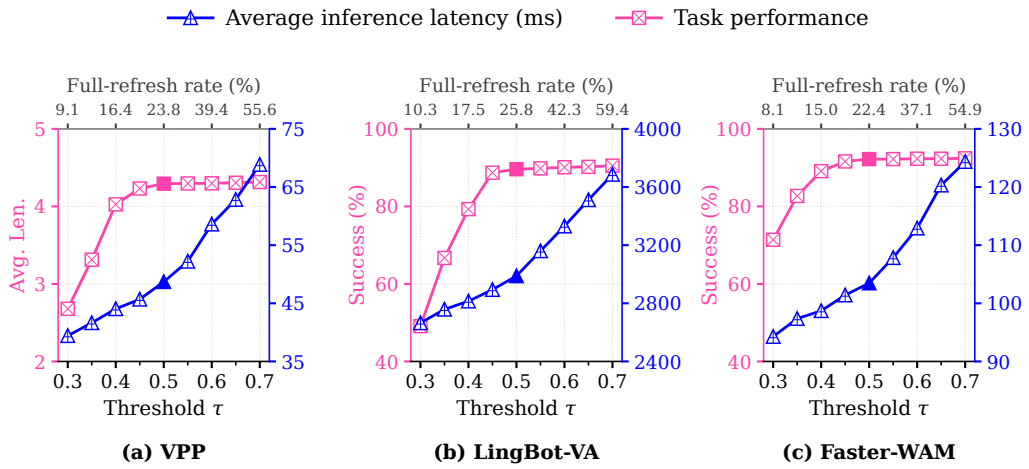


Figure 11: **Confidence threshold sensitivity.** Task performance (pink, left axes) and average inference latency (blue, right axes) versus τ for VPP on CALVIN and LingBot-VA and Faster-WAM on RoboTwin 2.0 Easy. Top labels give Full-refresh rates at the corresponding thresholds; Light accounts for the remainder.