REALTIMEWAM: ONE-STEP ASYNCHRONOUS WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) incorporate visual representations from video generation backbones to guide action prediction. Recent efficient WAMs adopt Mixture-of-Transformers (MoT) architectures and compute video representations once for reuse by the action expert. However, intra-expert iteration (*i.e.*, multi-step action denoising) and inter-expert waiting (*i.e.*, sequential execution of the video and action experts) still limit inference efficiency. To this end, we present RealtimeWAM, an extremely efficient WAM variant with one-step action generation and asynchronous inference, addressing these two bottlenecks. To reduce intra-expert iteration, we propose Teacher-Anchored Consistency Distillation (TACD) to address a local-global error gap: low local consistency error alone does not guarantee accurate final actions. TACD supplements local consistency with explicit supervision from the frozen teacher's multi-step rollout endpoint, enabling accurate one-step action generation. Additionally, we propose Cross-Expert Wavefront Pipelining (CEWP) to eliminate unnecessary expert-level waiting. It overlaps the two experts through block-wise sharing of the video KV cache, synchronizing only immediately before the corresponding action attention consumes it. Extensive experiments across diverse benchmarks (*e.g.*, LIBERO, LIBERO-Plus and RoboTwin) and model variants (*e.g.*, Fast-WAM and Faster-WAM) demonstrate the superiority of RealtimeWAM. Notably, RealtimeWAM maintains near-lossless performance (*i.e.*, $< 1\%$ drop) across these benchmarks while delivering significant end-to-end speedup (*e.g.*, $\sim 25\times$ on H100).

1 INTRODUCTION

World Action Models (WAMs) (Yuan et al., 2026; Ye et al., 2026; Bi et al., 2026; Li et al., 2026a; Kim et al., 2026; Zhao et al., 2026) have emerged as a promising paradigm for robot manipulation by jointly modeling future visual observations and robot actions. Unlike conventional Vision-Language-Action (VLA) models (Shukor et al., 2025; Intelligence et al., 2025; Kim et al., 2024; Bjorck et al., 2025; Black et al., 2024; Liu et al., 2025) that map visual observations and language instructions directly to actions, WAMs incorporate future visual prediction as an additional learning objective. This supervision encourages policies to capture the physical dynamics and temporal structure of robot–environment interactions.

Based on whether video and action modeling use the same network, WAM architectures can be broadly divided into two categories. Representative shared-backbone WAMs (Kim et al., 2026; Ye et al., 2026) jointly denoise video and action tokens within a single backbone. Consequently, iterative denoising of long token sequences through the heavy video backbone increases inference latency. In contrast, some efficient Mixture-of-Transformers (MoT)-based WAMs (Yuan et al., 2026; Zhao et al., 2026) offer a promising alternative: the video expert computes representations *once*, while a separate action expert reuses them for action generation.

Despite advancements in inference pipeline design, these MoT-based WAMs still incur substantial latency. Specifically, even on advanced GPUs (*e.g.*, H100 in Fig. 1), these models (*e.g.*, Fast-WAM (Yuan et al., 2026)) require over 300 ms for a single inference call. We identify two inference bottlenecks that remain insufficiently addressed in existing MoT-based WAMs: ① *Intra-expert iteration* arises from multi-step action denoising. We observe that the action expert accounts for nearly 90% of end-to-end latency for action generation across diverse GPUs (see Fig. 1). ② *Inter-expert*

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

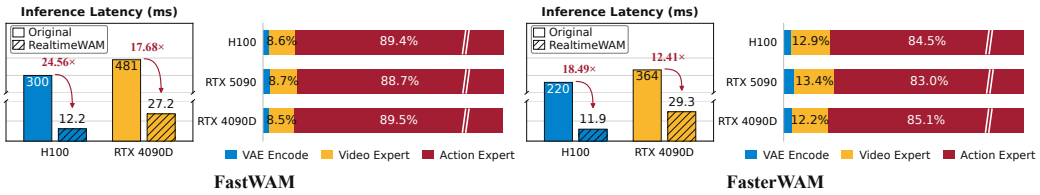


Figure 1: Inference efficiency of Fast-WAM (*left*) and Faster-WAM (*right*) across diverse hardware. Vertical bars compare inference latency between the original models and RealtimeWAM, while stacked bars show the runtime proportions of key components.

waiting stems from sequential execution of the video and action experts. As shown in Fig. 4(a), the action expert begins execution only after the video expert completes its forward pass, causing unnecessary waiting.

Motivated by the above analysis, we present *RealtimeWAM*, an extremely efficient post-trained variant based on *any* MoT-based WAM. To our knowledge, it is the *first* one-step WAM to achieve near-lossless performance compared with its multi-step counterparts.

To combat intra-expert iteration, we propose *Teacher-Anchored Consistency Distillation* (TACD) for one-step action generation. TACD addresses a *local-global error gap* in conventional consistency distillation, which only constrains local consistency while neglecting to explicitly penalize the global target action error. TACD adopts the frozen teacher's multi-step rollout endpoint as a numerical proxy for the exact clean endpoint, explicitly aligning the student's velocity prediction with the corresponding interval-averaged teacher velocity. From a theoretical perspective, we show that this teacher anchor directly constrains the total endpoint error, complementing local consistency supervision with an explicit action endpoint constraint.

To reduce inter-expert waiting, we introduce *Cross-Expert Wavefront Pipelining* (CEWP), which transforms coarse expert-wise execution into a fine-grained block-wise pipeline. Under conventional expert-wise execution, the Action Expert waits for the Video Expert to construct the complete block-wise KV cache. We first conduct an in-depth dependency analysis, observing that each action block requires the video KV cache only from its corresponding video block, rather than the complete output of the Video Expert. Motivated by this observation, CEWP overlaps the execution of the two experts through block-wise sharing of the video KV cache, synchronizing only immediately before the corresponding action attention consumes it, which significantly reduces unnecessary waiting.

RealtimeWAM achieves impressive results on representative MoT-based WAM architectures, including Fast-WAM (Yuan et al., 2026) and Faster-WAM (Zhao et al., 2026), across diverse downstream benchmarks (*e.g.*, LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2025), and RoboTwin 2.0 (Chen et al., 2025)). On these benchmarks, it delivers approximately $18 \sim 25\times$ inference acceleration with an average $< 1\%$ accuracy drop compared with the original multi-step models. Notably, RealtimeWAM's per-call inference latency is significantly shorter than the corresponding execution time (*e.g.*, 30 Hz control frequency), enabling real-time action generation. These results demonstrate that RealtimeWAM remains effective across diverse robot configurations, task horizons, and distribution conditions, and provide a practical path toward efficient WAM deployment.

To summarize, our contributions are as follows:

- ▶ We propose RealtimeWAM, a one-step asynchronous WAM that addresses both intra-expert iteration and inter-expert waiting in *any* MoT-based WAM for efficient robot manipulation.
- ▶ We propose *Teacher-Anchored Consistency Distillation* (TACD), which mitigates the *local-global error gap* by augmenting local consistency supervision with a teacher endpoint constraint for accurate one-step action generation.
- ▶ We introduce *Cross-Expert Wavefront Pipelining* (CEWP), which reduces unnecessary synchronization by overlapping block-wise execution of the Video and Action Experts according to their data dependencies.
- ▶ Extensive experiments demonstrate that RealtimeWAM achieves real-time action generation with $< 1\%$ accuracy degradation (*i.e.*, 12.2 ms inference latency with $\sim 25\times$ speedup on an NVIDIA H100 GPU).

2 Related Work

2.1 World Action Models

While Vision-Language-Action (VLA) models (Shukor et al., 2025; Intelligence et al., 2025; Kim et al., 2024; Bjorck et al., 2025; Black et al., 2024; Liu et al., 2025; Wang et al., 2026b; Zheng et al., 2026) rely on pretrained vision-language backbones to map visual observations and language instructions directly to actions, World Action Models (WAMs) (Yuan et al., 2026; Ye et al., 2026; Bi et al., 2026; Li et al., 2026a) incorporate future visual representations from video generation backbones to guide action prediction. Modern WAMs can be broadly classified into two categories. ① One line of work focuses on shared-backbone WAMs, which model future video and robot actions within a shared backbone, including DreamZero (Ye et al., 2026) and Cosmos (Kim et al., 2026). ② In contrast, Mixture-of-Transformers (MoT)-based WAMs, such as Motus (Bi et al., 2026), employ separate video and action experts that interact through attention. Among these methods, Fast-WAM (Yuan et al., 2026) reduces inference overhead by computing video representations *once* for reuse by the action expert. Subsequent works, including Light-WAM (Li et al., 2026c) and Faster-WAM (Zhao et al., 2026), further improve computational efficiency and out-of-distribution (OOD) robustness. Nevertheless, these methods may incur substantial performance degradation on challenging benchmarks or achieve limited speedups.

2.2 Few-Step Distillation

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) and flow models (Lipman et al., 2022; Liu et al., 2022; Albergo et al., 2023) have become cornerstones of modern high-quality image and video generation. However, their iterative sampling cost has motivated a series of few-step acceleration methods, including progressive distillation (Salimans & Ho, 2022), consistency models (Song et al., 2023), distribution matching distillation (Yin et al., 2024), MeanFlow (Geng et al., 2025), and adversarial distillation (Sauer et al., 2024). These distillation techniques have recently been introduced into World Action Models (Akbari et al., 2026; Yang et al., 2026). The work most relevant to ours is Flash-WAM (Akbari et al., 2026), which identifies a modality-dependent noise-regime mismatch and introduces modality-specific consistency parameterizations. However, ① it focuses on few-step distillation for WAMs that retain future-video denoising during inference, such as LingBot-VA (Li et al., 2026a); and ② its ultra-few-step (*i.e.*, one-step) distillation still exhibits noticeable performance degradation (*e.g.*, RoboTwin and real-world tasks). These limitations motivate our investigation of one-step action distillation for MoT-based WAMs.

2.3 Asynchronous Inference for Robot Policies

Asynchronous inference has been widely studied for real-time robot control because it mitigates execution stalls, stale observations, and delayed responses caused by policy latency. Existing approaches can be broadly grouped into three categories: execution-level asynchrony, module-level asynchrony, and sampling-level asynchrony. Execution-level methods overlap prediction of the next action chunk with execution of the current one and maintain continuity through prefix conditioning, state correction, or chunk fusion (Black et al., 2025a;b; Tang et al., 2025; Wang et al., 2026a; Motubrain Team, 2026). Module-level methods decouple a slow semantic or world module from a fast action module, allowing cached high-level representations to guide high-frequency action updates (Hirose et al., 2026; Cai et al., 2026). Sampling-level methods assign different action positions, chunks, modalities, or denoising streams distinct sampling schedules, enabling early action output or selective refinement without waiting for full decoding (Jiang et al., 2025; Lu et al., 2026; Li et al., 2026b). In contrast, RealtimeWAM employs finer-grained asynchrony at the block level, distinguishing it from prior approaches.

3 Preliminaries

3.1 Efficient Mixture-of-Transformers (MoT)-based WAMs

WAMs with a Mixture-of-Transformers (MoT) architecture (Bi et al., 2026; Li et al., 2026a) comprise a pretrained Video Expert for world modeling and an Action Expert for action generation.

Corresponding blocks in the two experts interact through shared attention. While the two experts are jointly trained on video and action prediction, recent efficient MoT-based WAMs (Yuan et al., 2026; Zhao et al., 2026) require only a single forward pass through the Video Expert during inference. Specifically, the Video Expert processes the current observation o and language instruction l to extract block-wise keys and values:

$$z(o, l) = F_{\psi}^{\mathrm{KV}}(o, l) = \{(K_i^v, V_i^v)\}_{i=1}^{N}. \tag{1}$$

Here, F_{ψ}^{KV} denotes the KV extraction function of the Video Expert with parameters ψ. The latent world representation $z(o, l)$ consists of keys K_i^v and values V_i^v from each of the N video transformer blocks. Subsequently, these efficient MoT-based WAMs use this representation to parameterize the action distribution:

$$p_\theta(a \mid o, l) = p_\theta(a \mid z(o, l)). \tag{2}$$

Here, a denotes an action chunk with prediction horizon H, and θ denotes the parameters of the Action Expert. Action generation requires n denoising steps along $0 = t_0 < \cdots < t_n = 1$. The action distribution $p_\theta(a \mid z(o, l))$ is implicitly defined by the denoising map $\Phi_\theta^z = \Phi_{\theta,t_1}^z \circ \Phi_{\theta,t_2}^z \circ \cdots \circ \Phi_{\theta,t_n}^z$, where $a_{t_{j-1}} = \Phi_{\theta,t_j}^z(a_{t_j}) = a_{t_j} - (t_j - t_{j-1}) v_\theta(a_{t_j}, t_j)$. Here, a_{t_j} denotes a noisy action chunk and v_θ the predicted velocity.

3.2 Consistency Distillation for Action Generation

Consistency distillation (CD) (Song et al., 2023) learns to map noisy states (*i.e.*, action chunks in our paper) along the same teacher denoising trajectory to a common clean endpoint (*i.e.*, $t = 0$), enabling few-step generation. It uses three models: a frozen teacher θ_{T}, a trainable student θ_{S}, and an exponential moving average (EMA) target θ_{ema}. During training, given a clean action chunk a_0^{data} from the training data, we randomly sample $t \in [0, 1]$ and Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ to construct $a_t = (1 - t) a_0^{\mathrm{data}} + t\epsilon$. The student and EMA target use the endpoint mapping

$$f_\theta(a_t, t) = a_t - t\, v_\theta(a_t, t). \tag{3}$$

For a neighboring time $0 \le s < t$, the teacher produces

$$\tilde{a}_s = \texttt{Solver}(a_t, t, s; \theta_{\mathrm{T}}) \approx a_t - (t - s)\, v_{\theta_{\mathrm{T}}}(a_t, t), \tag{4}$$

where $\texttt{Solver}(a_t, t, s; \theta_{\mathrm{T}})$ denotes one numerical integration step from t to s using the frozen teacher θ_{T}. For neighboring time steps with small $t - s$, treating the velocity as locally constant yields the first-order approximation above. The student is optimized with

$$\mathcal{L}_{\mathrm{CD}} = \mathbb{E}_{a_t, t, s}\left[\| f_{\theta_{\mathrm{S}}}(a_t, t) - \mathrm{sg}[f_{\theta_{\mathrm{ema}}}(\tilde{a}_s, s)] \|_2^2 \right], \tag{5}$$

where sg denotes stop-gradient. After each student update, $\theta_{\mathrm{ema}} \leftarrow \mu\theta_{\mathrm{ema}} + (1 - \mu)\theta_{\mathrm{S}}$, with decay rate $\mu \in [0, 1)$. Therefore, consistency distillation learns to map noisy action chunks directly to clean endpoints, enabling few-step action generation without traversing the full teacher trajectory.

4 RealtimeWAM

In this work, we propose *RealtimeWAM*, a one-step asynchronous world action model for post-training acceleration of *any* MoT-based WAM. RealtimeWAM addresses intra-expert iteration through Teacher-Anchored Consistency Distillation (Sec. 4.1) and inter-expert waiting through Cross-Expert Wavefront Pipelining (Sec. 4.2).

4.1 Teacher-Anchored Consistency Distillation

As discussed above, multi-step action denoising dominates inference latency (Fig. 1) while video latent representations are computed in a single forward pass in the Video Expert. Therefore, our target is to achieve high-quality few-step action generation (especially one-step) to combat intra-expert iteration. Recent step distillation strategies (Song et al., 2023; Yin et al., 2024; Geng et al., 2025; Huang et al., 2026) have been extensively studied in image and video generation. However, we identify a distinctive characteristic of action generation in WAMs: denoising is strongly conditioned on visual observations, robot states, and language instructions, which can confine feasible actions (*e.g.*, grasp positions and gripper commands) to a narrow range. Within this constrained action space, even subtle deviations can affect task success rates owing to the high sensitivity of robotic control to action errors.

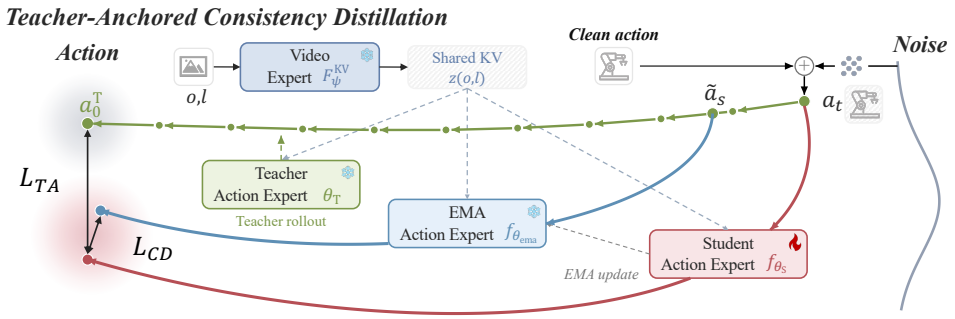


Figure 2: Overview of Teacher-Anchored Consistency Distillation (TACD). The frozen Video Expert provides shared KV conditioning, while the action student is trained with local consistency supervision and a teacher-anchored constraint derived from the frozen teacher's multi-step rollout.

However, directly applying consistency distillation to WAMs raises a significant challenge, dubbed the *local-global error gap*. Specifically, although consistency distillation (Song et al., 2023) can effectively minimize *local* consistency errors, it does not considerably reduce deviations from the *global* target action. As shown in Fig. 3 (right), the error between the teacher's clean action targets and student's predictions does not decrease considerably for consistency distillation. Such large deviations from the clean action targets can induce inaccuracies in robotic control.

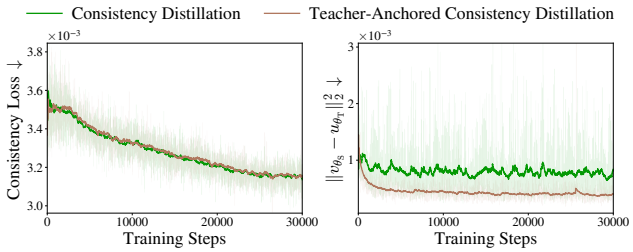


Figure 3: Training curves of two distillation methods: consistency loss (*Left*) and squared distance between the student's velocity v_{θ_S} and the teacher's average velocity towards the clean endpoint u_{θ_T} (*Right*).

Formally, let a_t denote a noisy action chunk and $\tilde{a}_s$ the teacher-updated action defined analogously to Eq. (4). Let $a_0^\star$ denote the exact clean endpoint of the frozen teacher ODE initialized at a_t, where $\star$ indicates the theoretical exact solution. Omitting the shared conditioning $z(o,l)$ for brevity, the *total* error decomposes into a *local* consistency error and a *global* target error:

$$\underbrace{f_{\theta_S}(a_t,t) - a_0^\star}_{\mathbf{e}_{\text{total}}:\ \text{Total error}} = \underbrace{f_{\theta_S}(a_t,t) - f_{\theta_{\text{ema}}}(\tilde{a}_s,s)}_{\mathbf{e}_{\text{local}}:\ \text{Local consistency error}} + \underbrace{f_{\theta_{\text{ema}}}(\tilde{a}_s,s) - a_0^\star}_{\mathbf{e}_{\text{global}}:\ \text{Global target error}} . \tag{6}$$

Consistency Distillation directly minimizes $\mathbb{E}\|\mathbf{e}_{\text{local}}\|_2^2$ but does not explicitly penalize $\mathbf{e}_{\text{global}}$. Consequently, a small *local* consistency error alone does not guarantee a small total error $\mathbf{e}_{\text{total}}$ (see Appendix C.1). Therefore, our key insight is to preserve *local* consistency supervision while introducing an additional constraint based on the teacher's target action (*i.e.*, $a_0^\star$), thereby mitigating the *global* deviation that the local objective may leave unresolved.

Motivated by this insight, we propose *Teacher-Anchored Consistency Distillation* (TACD), a novel post-training step distillation framework tailored for MoT-based WAMs. Specifically, since the exact endpoint $a_0^\star$ is unavailable in practice, we adopt the frozen teacher's multi-step rollout endpoint a_0^{T} as a proxy and compute the corresponding interval-averaged velocity. It is worth noting that the three Action Experts (*i.e.*, the frozen teacher, trainable student, and EMA target) share the same video KV cache $z(o,l)$ from a single frozen Video Expert. Starting from the same noisy action a_t under this shared conditioning $z(o,l)$, we have

$$a_0^{\text{T}} = \texttt{Solver}(a_t,t,0;\theta_{\text{T}}), \qquad u_{\theta_{\text{T}}}(a_t,t) = \frac{a_t - a_0^{\text{T}}}{t}, \quad t>0, \tag{7}$$

where $\texttt{Solver}$ denotes numerical integration from t to 0 using K teacher denoising steps.

We further directly supervise the student's velocity prediction $v_{\theta_S}(a_t,t)$ with the teacher's interval-averaged velocity $u_{\theta_T}(a_t,t)$, yielding the teacher-anchored loss and the overall training objective:

$$\mathcal{L}_{\text{TA}} = \mathbb{E}_{a_t,t}\left[\|v_{\theta_S}(a_t,t) - \text{sg}[u_{\theta_T}(a_t,t)]\|_2^2\right], \qquad \mathcal{L} = \mathcal{L}_{\text{CD}} + \lambda\mathcal{L}_{\text{TA}}, \tag{8}$$

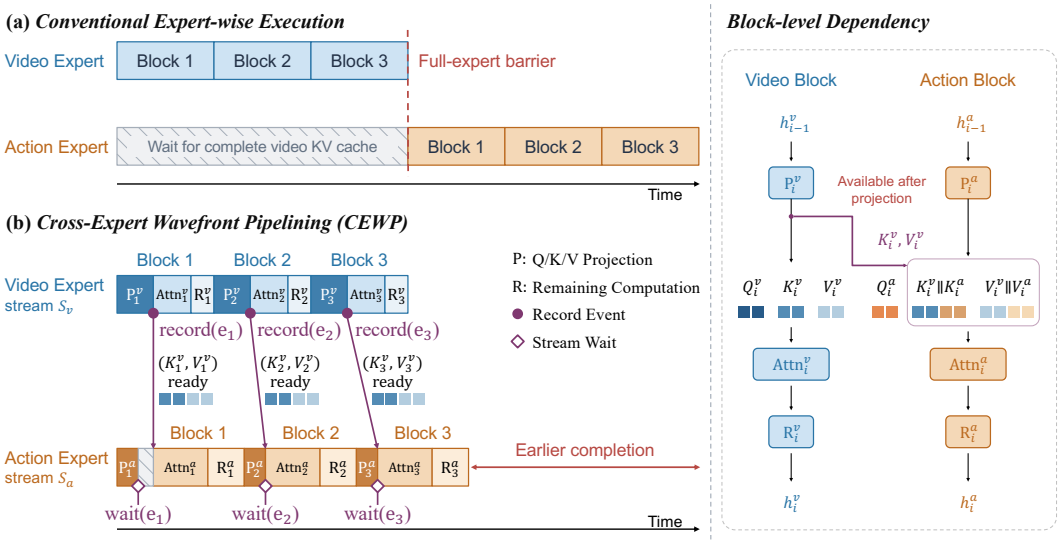


Figure 4: Comparison of expert execution schedules. (a) Conventional execution waits for the complete video KV cache. (b) Cross-Expert Wavefront Pipelining releases block-wise KV cache entries after projection and synchronizes only before action attention consumes them, overlapping the Video and Action Experts.

where $\lambda > 0$ weights the teacher-anchored loss. Equivalently, $\mathcal{L}_{\text{TA}}$ directly supervises the student's clean action prediction against the teacher's rollout endpoint, with weight $1/t^2$ (see Appendix C.2).

To relate this supervision to the exact teacher endpoint, let $\boldsymbol{\delta} = a_0^{\text{T}} - a_0^{\star}$ denote the teacher's numerical integration error. The total endpoint error in Eq. (6) then satisfies

$$\mathbf{e}_{\text{total}} = \mathbf{e}_{\text{local}} + \mathbf{e}_{\text{global}} = -t[v_{\theta_{\text{S}}}(a_t, t) - u_{\theta_{\text{T}}}(a_t, t)] + \boldsymbol{\delta}. \tag{9}$$

Consequently, teacher anchoring directly constrains the combined error $\mathbf{e}_{\text{local}} + \mathbf{e}_{\text{global}}$, complementing the local consistency objective with a direct endpoint constraint[1]. In addition, for $0 < t \leq 1$, this relation yields an upper bound on the expected squared endpoint error, $\mathbb{E}\|\mathbf{e}_{\text{total}}\|_2^2 \leq 2\mathcal{L}_{\text{TA}} + 2\mathbb{E}\|\boldsymbol{\delta}\|_2^2$ (see Appendix C.3).

4.2 Cross-Expert Wavefront Pipelining

Teacher-Anchored Consistency Distillation reduces intra-expert iteration by enabling one-step action generation. However, inter-expert waiting persists because the Video and Action Experts still execute sequentially (Fig. 4(a)). Specifically, the Action Expert waits for the complete video KV cache, creating a global synchronization barrier that prevents action computation from starting even when its required context is already available.

We therefore analyze the actual data dependencies between the Video Expert and Action Expert in a fine-grained manner. We first decompose each DiT block into three sequential components: Q/K/V projection $\mathcal{P}_i$, attention Attn_i, and the remaining computation $\mathcal{R}_i$. Let h_{i-1}^v and h_{i-1}^a denote the video and action hidden states entering block $i \in \{1, \ldots, N\}$, respectively. Their projections are

$$(Q_i^v, K_i^v, V_i^v) = \mathcal{P}_i^v(h_{i-1}^v), \qquad (Q_i^a, K_i^a, V_i^a) = \mathcal{P}_i^a(h_{i-1}^a). \tag{10}$$

Here, (K_i^v, V_i^v) constitute the i-th entry of $z(o, l)$ in Eq. (1). The Video Expert attends to its own tokens, while the Action Expert additionally incorporates the corresponding video KV cache:

$$\begin{aligned} h_i^v &= \mathcal{R}_i^v\big(h_{i-1}^v, \text{Attn}_i^v(Q_i^v, K_i^v, V_i^v)\big), \\ h_i^a &= \mathcal{R}_i^a\big(h_{i-1}^a, \text{Attn}_i^a(Q_i^a, K_i^v \parallel K_i^a, V_i^v \parallel V_i^a)\big), \end{aligned} \tag{11}$$

where $\parallel$ denotes token-wise concatenation. Action attention Attn_i^a is thus the only operation in action block i that directly consumes the video KV cache (K_i^v, V_i^v), with the dependency

$$\mathcal{P}_i^v \longrightarrow \text{Attn}_i^a \longleftarrow \mathcal{P}_i^a. \tag{12}$$

[1]The total endpoint error $\mathbf{e}_{\text{total}}$ also depends on the teacher's numerical integration error $\boldsymbol{\delta}$, which can be controlled by increasing the number of teacher denoising steps K (see Appendix C.3).

Our key observations are: ① The action projection $\mathcal{P}_i^a$ can execute once h_{i-1}^a is available, without waiting for the video KV cache (K_i^v, V_i^v). ② Action attention Attn_i^a can begin once both $\mathcal{P}_i^a$ and $\mathcal{P}_i^v$ complete, without waiting for video attention Attn_i^v or video states h_j^v with $j \geq i$. ③ Existing methods (Yuan et al., 2026; Zhao et al., 2026) enforce $\mathcal{R}_N^v \rightarrow \mathcal{P}_1^a$, introducing unnecessary waiting.

Based on this dependency analysis, we propose *Cross-Expert Wavefront Pipelining* (CEWP), which transforms coarse expert-wise execution into a fine-grained block-wise wavefront across the Video and Action Experts. We assign the Video and Action Experts to two non-blocking CUDA streams, $\mathcal{S}_v$ and $\mathcal{S}_a$, respectively. Each stream preserves program order, while a CUDA event e_i enforces the dependency at block i:

$$\begin{aligned} \mathcal{S}_v &: \mathcal{P}_i^v \rightarrow \mathrm{record}(e_i) \rightarrow \mathrm{Attn}_i^v \rightarrow \mathcal{R}_i^v, \\ \mathcal{S}_a &: \mathcal{P}_i^a \rightarrow \mathrm{wait}(e_i) \rightarrow \mathrm{Attn}_i^a \rightarrow \mathcal{R}_i^a, \end{aligned} \tag{13}$$

where e_i signals the availability of (K_i^v, V_i^v). The video stream records e_i immediately after $\mathcal{P}_i^v$ and continues its remaining computation, while the action stream waits only before Attn_i^a (shown in Fig. 4(b)). This late synchronization allows the action projection to proceed independently and delays cross-stream waiting until the video KV cache is actually consumed. Action blocks thus consume video KV cache entries incrementally as the video stream advances, forming a staggered wavefront along model depth.

5 Experiments

5.1 Implementation Details

We implement RealtimeWAM based on Fast-WAM (Yuan et al., 2026) and Faster-WAM (Zhao et al., 2026), yielding RealtimeWAM* and RealtimeWAM†, respectively. We freeze the Video Expert and optimize only rank-128 low-rank adaptation (LoRA) (Hu et al., 2021) parameters in the Action Expert. In Teacher-Anchored Consistency Distillation (TACD), we obtain the teacher endpoint through a 10-step rollout of the frozen teacher and set the teacher-anchored loss weight to $\lambda = 0.2$. Training proceeds for 30,000 iterations with AdamW, a learning rate of 10^{-4}, and a weight decay of 10^{-2}. We set the exponential moving average (EMA) decay to 0.995. For a fair comparison, all evaluated step-distillation methods, including distribution matching distillation (DMD) (Yin et al., 2024), consistency distillation (Song et al., 2023), and MeanFlow distillation (Geng et al., 2025; Gu et al., 2026), adopt the same training settings. Moreover, we compare against other efficient WAMs, including Flash-WAM (Akbari et al., 2026) and Light-WAM (Li et al., 2026c). For latency evaluation, we include VAE encoding time but exclude text encoding time, as instructions are encoded only once per episode.

5.2 Evaluation Benchmarks

We evaluate RealtimeWAM on LIBERO (Liu et al., 2023), RoboTwin 2.0 (Chen et al., 2025), and LIBERO-Plus (Fei et al., 2025), using task success rate as the evaluation metric. On LIBERO, we evaluate four task suites: Spatial, Object, Goal, and Long. For RoboTwin 2.0, we follow the multi-task training setting in previous work (Yuan et al., 2026), training on 50 tasks using a total of 2,500 clean demonstrations and 25,000 randomized demonstrations. We evaluate performance under both clean and randomized conditions. To assess out-of-distribution (OOD) robustness, we further evaluate success rates on all seven perturbation subsets of LIBERO-Plus. We additionally evaluate RealtimeWAM on real-world dual-arm manipulation tasks (see Appendix F.6).

5.3 Main Results

RoboTwin 2.0. We report detailed results on RoboTwin 2.0 (Tab. 1). Notably, RealtimeWAM* achieves a success rate of 90.84% with one-step action generation using only LoRA fine-tuning, incurring a reduction of just 0.67% compared with 10-step Fast-WAM. Surprisingly, RealtimeWAM† achieves nearly lossless performance with one-step action generation, attaining a success rate of 92.64% with a reduction of only 0.29% compared with 10-step Faster-WAM. It also considerably outperforms other efficient WAMs (*e.g.*, 81.41% for one-step Flash-WAM) and step-distillation baselines (*e.g.*, 88.22% for DMD*).

Table 1: Success rates (%) on RoboTwin 2.0 and LIBERO. NFE denotes the number of denoising steps for video and action generation. Methods marked with * and † are built on Fast-WAM and Faster-WAM, respectively.

Method	NFE		RoboTwin 2.0			LIBERO
	Video	Action	Clean↑	Random↑	Overall↑	Overall↑
WAM and VLA Baselines						
π_0 (Black et al., 2024)	–	–	65.92	58.40	62.16	94.1
$\pi_{0.5}$ (Intelligence et al., 2025)	–	–	82.74	76.76	79.75	96.9
X-VLA (Zheng et al., 2026)	–	–	72.90	72.80	72.85	98.1
Motus (Bi et al., 2026)	10	10	88.66	87.02	87.84	97.7
LingBot-VA (Li et al., 2026a)	25	50	92.90	91.50	92.20	98.5
Fast-WAM (Yuan et al., 2026)	1	10	91.82	91.19	91.51	97.0
Faster-WAM (Zhao et al., 2026)	1	10	93.20	92.66	92.93	98.9
Other Efficient Variants						
Flash-WAM (Akbari et al., 2026)	1	2	88.42	82.66	85.54	96.1
Flash-WAM (Akbari et al., 2026)	1	1	82.56	80.26	81.41	95.1
Light-WAM (Li et al., 2026c)	1	1	76.40	76.30	76.40	97.2
Few-Step Distillation						
Fast-WAM (Yuan et al., 2026)	1	1	70.42	68.12	69.27	95.0
Mean Flow* (Geng et al., 2025)	1	1	85.74	85.72	85.73	96.9
Consistency Distillation* (Song et al., 2023)	1	1	89.88	89.38	89.63	96.9
DMD* (Yin et al., 2024)	1	1	88.64	87.80	88.22	96.1
RealtimeWAM*	1	1	91.96	89.72	90.84	97.0
RealtimeWAM†	1	1	92.98	92.30	92.64	99.0

Table 2: Success rates (%) on the seven perturbation subsets of LIBERO-Plus.

Method	Camera↑	Robot↑	Lang.↑	Light↑	Backg.↑	Noise↑	Layout↑	Overall↑
UniVLA (Bu et al., 2025)	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
OpenVLA-OFT (Kim et al., 2025)	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
π_0 (Black et al., 2024)	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
π_0-Fast (Pertsch et al., 2025)	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
WorldVLA (Cen et al., 2025)	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
Fast-WAM (Yuan et al., 2026)	18.8	45.7	70.1	83.2	45.7	29.8	62.7	49.1
Faster-WAM (Zhao et al., 2026)	53.8	71.6	94.7	96.3	61.3	63.6	79.1	73.6
RealtimeWAM†	54.2	69.0	92.5	94.7	59.4	64.6	79.3	73.0

LIBERO. A similar trend is observed on LIBERO (Tab. 1). RealtimeWAM* achieves an overall success rate of 97.0% with one-step action generation, matching the performance of 10-step Fast-WAM. Compared with directly reducing Fast-WAM to a single denoising step (95.0%), RealtimeWAM* improves the success rate by 2.0%, demonstrating the effectiveness of step distillation in preserving policy performance. It also outperforms one-step Flash-WAM (95.1%) and DMD* (96.1%), while remaining competitive with Light-WAM (97.2%). Detailed results for individual LIBERO task suites are provided in Appendix F.3 (Tab. F.1).

LIBERO-Plus. We further evaluate robustness to distribution shifts on LIBERO-Plus (Tab. 2). With one-step action generation, RealtimeWAM† achieves an overall success rate of 73.0%, only 0.6% below Faster-WAM (73.6%). It slightly improves performance under camera, sensor-noise, and layout perturbations, indicating that step distillation largely preserves out-of-distribution robustness.

5.4 Ablation Study

Effect of different components. We first validate the effectiveness of the key components of our distillation framework (Tab. 3). Fine-tuning the Video Expert during action step distillation reduces the overall success rate while introducing additional training overhead (*i.e.*, longer training time and higher peak memory usage, as reported in Appendix A). With the Video Expert frozen, augmenting

Table 3: Ablation of loss terms and expert fine-tuning on RoboTwin 2.0. $\mathcal{L}_{\text{CD}}$ and $\mathcal{L}_{\text{TA}}$ denote the consistency and teacher-anchored losses, respectively. Video/Action indicate whether each expert is fine-tuned. Results are success rates (%); best results are in bold.

$\mathcal{L}_{\text{CD}}$	$\mathcal{L}_{\text{TA}}$	Video	Action	Clean↑	Random↑	Overall↑
✓	✗	✓	✓	89.88	89.38	89.63
✓	✗	✗	✓	89.76	**89.94**	89.85
✓	✓	✗	✓	**91.96**	89.72	**90.84**

Table 4: Effect of teacher rollout steps K in TACD on RoboTwin 2.0. K denotes the number of teacher denoising steps used to obtain a_0^{T}. We use $K = 10$ by default. Results are success rates (%); best results are in bold.

K	Clean↑	Random↑	Overall↑
5	91.02	89.76	90.39
10	**91.96**	89.72	**90.84**
20	91.22	**90.32**	90.77

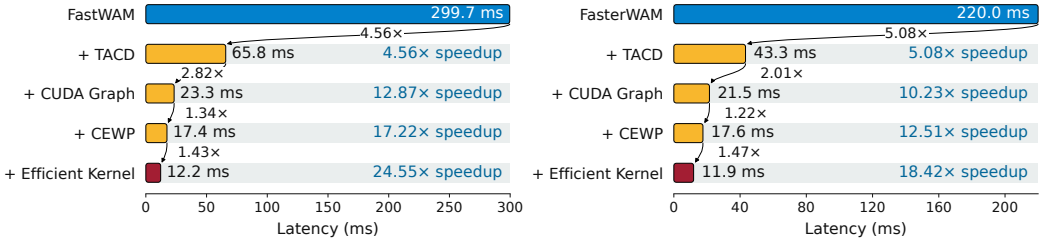


Figure 5: Inference latency of RealtimeWAM built on Fast-WAM (*left*) and Faster-WAM (*right*) on a single NVIDIA H100 GPU.

$\mathcal{L}_{\text{CD}}$ with $\mathcal{L}_{\text{TA}}$, yielding *Teacher-Anchored Consistency Distillation* (TACD), further improves the reported overall success rate from 89.85% to 90.84%. This improvement is consistent with our analysis: the teacher anchor provides an endpoint reference from the complete teacher trajectory, supplementing *local* consistency supervision with an explicit endpoint constraint to mitigate *global* deviations that the local objective may leave unresolved.

Choice of teacher rollout steps K. We further present results for different numbers of teacher rollout steps K (Tab. 4). With $K = 5$, the rollout may produce a less accurate approximation a_0^{T} to the exact clean endpoint $a_0^\star$, resulting in a larger numerical integration error $\boldsymbol{\delta}$ and weaker endpoint supervision. This setting achieves a success rate of 90.39%, which improves to 90.84% when K increases to 10. Further increasing K to 20 yields no additional improvement (90.77%) while requiring more teacher computation. We therefore adopt $K = 10$ to balance policy performance and teacher rollout cost.

Efficiency Results. RealtimeWAM reduces inference latency through algorithm and system co-optimization (Fig. 5). In addition to TACD and CEWP, we employ CUDA Graph and efficient kernels from LightX2V (LightX2V Contributors, 2025) (see Appendix F.2). On Fast-WAM and Faster-WAM, TACD reduces latency from 299.7 and 220.0 ms to 65.8 and 43.3 ms, yielding $4.56\times$ and $5.08\times$ speedups, respectively. After enabling CUDA Graph, CEWP further reduces latency from 23.3 to 17.4 ms on Fast-WAM and from 21.5 to 17.6 ms on Faster-WAM, providing additional speedups of $1.34\times$ and $1.22\times$. With all optimizations enabled, RealtimeWAM achieves inference latencies of 12.2 and 11.9 ms, corresponding to overall speedups of $24.55\times$ and $18.42\times$ over the respective baselines. Detailed latency results on other GPUs (*e.g.*, RTX 4090D and RTX 5090) are provided in Appendix F.5 (Tab. F.3).

6 Conclusion

We presented RealtimeWAM, a post-training framework that accelerates MoT-based WAMs by addressing intra-expert iteration and inter-expert waiting. *Teacher-Anchored Consistency Distillation* complements local consistency supervision with a teacher endpoint constraint, enabling accurate one-step action generation. *Cross-Expert Wavefront Pipelining* overlaps block-wise execution of the Video and Action Experts through fine-grained synchronization, reducing unnecessary waiting. RealtimeWAM enables real-time action generation (*e.g.*, 12.2 ms latency) with near-lossless policy performance on MoT-based WAMs, including Fast-WAM and Faster-WAM, while delivering significant end-to-end speedups across diverse hardwares (*e.g.*, approximately $18 \sim 25\times$ on H100).

AI USE STATEMENT

In this work, we used generative AI tools solely to improve the language and readability of the manuscript. We did not use these tools for any tasks requiring disclosure under the ICLR 2027 AI Policy for Authors. The authors reviewed all AI-assisted edits to ensure that they preserved the original technical meaning. We take responsibility for the final content of this work, including the AI-assisted language edits.

REFERENCES

Arman Akbari, Ci Zhang, Arash Akbari, Lin Zhao, Yixiao Chen, Weiwei Chen, Xuan Zhang, Geng Yuan, and Yanzhi Wang. Flash-wam: Modality-aware distillation for world action models. *arXiv preprint arXiv:2606.05254*, 2026.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

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

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025a.

Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025b.

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

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, et al. Aha-wam: Asynchronous horizon-adaptive world-action modeling with observation-guided context routing. *arXiv preprint arXiv:2606.09811*, 2026.

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

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

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. Anyflow: Any-step video diffusion model with on-policy flow map distillation. In *European Conference on Computer Vision*, pp. 163–181. Springer, 2026.

E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations I: Nonstiff problems*. Springer, 1993.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Asyncvla: An asynchronous vla for fast and robust navigation on the edge. *arXiv preprint arXiv:2602.13476*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33: 6840–6851, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yushi Huang, Xiangxin Zhou, Jun Zhang, Liefeng Bo, and Tianyu Pang. Meanflownft: Bringing forward-process rl to average-velocity generators. *arXiv preprint arXiv:2607.15273*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Yuhua Jiang, Shuang Cheng, Yan Ding, Feifei Gao, and Biqing Qi. Asyncvla: Asynchronous flow matching for vision-language-action models. *arXiv preprint arXiv:2511.14148*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026a.

Zekai Li, Jiaming Tang, and Zhijian Liu. Flashvla: Streaming action decoding for fast and asynchronous vla inference. *arXiv preprint arXiv:2608.27384*, 2026b.

Ziang Li, Dongzhou Cheng, Yibin Wang, Shiyue Wang, Xiaoyang Xu, Lingxuan Weng, Juan Wang, and Jiaqi Wang. Light-wam: Efficient world action models with state-fusion action decoding. *arXiv preprint arXiv:2606.08242*, 2026c.

LightX2V Contributors. LightX2V: Light video generation inference framework. https://github.com/ModelTC/lightx2v, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. Faster: Rethinking real-time flow vlas. *arXiv preprint arXiv:2603.19199*, 2026.

Motubrain Team. World action models in real time: An empirical study of smooth execution via asynchronous deployment. *arXiv preprint arXiv:2608.01880*, 2026.

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

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pp. 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 32211–32252, 2023.

J Michael Steele. *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press, 2004.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. Vlash: Real-time vlas via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031*, 2025.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Yuzhang Shang, Ramana Rao Kompella, and Gaowen Liu. Real-time robot execution with masked action chunking. *arXiv preprint arXiv:2601.20130*, 2026a.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI conference on artificial intelligence*, volume 40, pp. 18638–18646, 2026b.

Liuhaichen Yang, Zhuang Jiang, Chenchao Sheng, and Zezhi Tang. Wam-opd: On-policy distillation for world action models. *arXiv preprint arXiv:2608.22364*, 2026.

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

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. *arXiv preprint arXiv:2405.14867*, 2024.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

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.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

APPENDIX

A Limitations **14**

B Cross-Expert Wavefront Pipelining: Pseudocode **15**

C Complete Error Derivation **16**

C.1 Local Consistency and Endpoint Error 16

C.2 Equivalence to Weighted Endpoint Supervision 16

C.3 Total Endpoint Error Bound 17

D Inference Equivalence of Asynchronous Execution **17**

E Inference Latency Analysis **17**

F More Experimental Details and Results **18**

F.1 Result sources and aggregation 18

F.2 Efficient Kernel Implementation 18

F.3 Detailed LIBERO Results 18

F.4 Additional Ablation Study 19

F.5 Latency on Additional GPUs 19

F.6 Real-World Evaluation of Long-Horizon Garment Folding 19

A LIMITATIONS

1) Applicability. RealtimeWAM targets MoT-based WAMs and is not directly applicable to shared-backbone architectures such as DreamZero (Ye et al., 2026). Step distillation in shared-backbone WAMs more closely resembles step distillation in video generation, making action-specific characteristics harder to isolate and motivating our focus on MoT architectures. Within this scope, our framework supports both models that omit explicit future-video prediction at inference (*e.g.*, Fast-WAM (Yuan et al., 2026)) and those that retain it (*e.g.*, Faster-WAM (Zhao et al., 2026)).

2) Training overhead. The teacher-anchored loss $\mathcal{L}_{\text{TA}}$ introduces additional training overhead by requiring multi-step rollouts from the frozen teacher. Nevertheless, this overhead remains manageable: as shown in Tab. A.1, adding $\mathcal{L}_{\text{TA}}$ increases training time from 5h 21min to 8h 3min (approximately 50%), comparable to jointly fine-tuning the Video and Action Experts (8h 1min). By freezing the Video Expert, TACD reduces peak GPU memory from 65.98 to 24.73 GiB compared with joint fine-tuning, a 62.5% reduction, while retaining the same memory footprint as action-only consistency distillation.

Table A.1: Training time and peak GPU memory for 30,000 training iterations on 16 NVIDIA H100 GPUs. $\mathcal{L}_{\text{CD}}$ and $\mathcal{L}_{\text{TA}}$ denote the consistency and teacher-anchored losses, respectively. Video/Action indicate whether each expert is fine-tuned.

$\mathcal{L}_{\text{CD}}$	$\mathcal{L}_{\text{TA}}$	Video	Action	Training Time↓	Peak Memory (GiB)↓
✓	✗	✓	✓	8h 1min	65.98
✓	✗	✗	✓	5h 21min	24.73
✓	✓	✗	✓	8h 3min	24.73

B Cross-Expert Wavefront Pipelining: Pseudocode

Our Cross-Expert Wavefront Pipelining (CEWP) supports MoT-based WAMs, including both models that omit explicit future-video prediction at inference (*e.g.*, Fast-WAM (Yuan et al., 2026)) and those that retain it (*e.g.*, Faster-WAM (Zhao et al., 2026)). Algorithm 1 presents a unified schedule for one paired forward pass. Input preparation, attention masks, output heads, and denoising schedules follow the base model.

Let $\mathcal{C}$ denote the video-conditioned action blocks, indexed from 1. Fast-WAM uses $\mathcal{C} = \{1, \ldots, N\}$, whereas our Faster-WAM implementation uses $\mathcal{C} = \{1, 5, \ldots, N\}$. At each conditioning block, `Fuse` returns the single-block KV for Fast-WAM or the base model's weighted fusion of the accumulated interval KV for Faster-WAM. The video stream then records a readiness event, which the action stream waits on only before attention. Intervening action blocks require no cross-expert synchronization.

Algorithm 1 Cross-Expert Wavefront Pipelining for MoT-based WAMs.

```
# hv: video states, including future tokens if used; ha: action states.
# P: Q/K/V projection, including key normalization and RoPE.
# R: remaining block operations; Mv, Ma: base-model masks.
def cewp(hv, ha, N, C, retain_video=False):
    caller = current_stream()
    Sv, Sa = Stream(), Stream()
    ready = {i: Event() for i in C}
    Sv.wait_stream(caller)
    Sa.wait_stream(caller)
    pending, cache = [], {}

    for i in range(1, N + 1):
        with stream(Sv):
            qv, kv, vv = P_v[i](hv)
            pending.append((kv, vv))
            if i in C:
                cache[i] = Fuse[i](pending)
                pending = []
                ready[i].record(Sv)  # Publish before video attention.
            hv = R_v[i](hv, Attn_v[i](qv, kv, vv, Mv[i]))

        with stream(Sa):
            qa, ka, va = P_a[i](ha)  # No video dependency yet.
            mask = None
            if i in C:
                Sa.wait_event(ready[i])  # Wait only at consumption.
                kc, vc = cache[i]
                kc.record_stream(Sa)
                vc.record_stream(Sa)
                ka, va = cat(kc, ka), cat(vc, va)
                mask = Ma[i]
            ha = R_a[i](ha, Attn_a[i](qa, ka, va, mask))

    caller.wait_stream(Sv)
    caller.wait_stream(Sa)
    ha.record_stream(caller)
    hv.record_stream(caller)
    action_pred = ActionHead(ha)
    video_pred = VideoHead(hv) if retain_video else None
    return action_pred, video_pred
```

C Complete Error Derivation

C.1 Local Consistency and Endpoint Error

We fix the model parameters and omit the shared conditioning $z(o, l)$. For $0 < t \leq 1$, let $a^\star_\tau$ be the unique teacher ODE solution with $a^\star_t = a_t$, and let $a^\star_0$ denote its exact endpoint. All expectations below use the same training sampling distribution, and we assume finite second moments for the errors and velocity residuals.

Adding and subtracting the EMA prediction makes the two error terms explicit:

$$\begin{aligned}\mathbf{e}_{\text{total}} &= f_{\theta_{\text{S}}}(a_t, t) - a^\star_0 \\ &= \underbrace{f_{\theta_{\text{S}}}(a_t, t) - f_{\theta_{\text{ema}}}(\tilde{a}_s, s)}_{\mathbf{e}_{\text{local}}} + \underbrace{f_{\theta_{\text{ema}}}(\tilde{a}_s, s) - a^\star_0}_{\mathbf{e}_{\text{global}}}.\end{aligned}$$

Expanding the squared norm and taking expectations gives

$$\mathbb{E}\|\mathbf{e}_{\text{total}}\|_2^2 = \mathbb{E}\|\mathbf{e}_{\text{local}}\|_2^2 + \mathbb{E}\|\mathbf{e}_{\text{global}}\|_2^2 + 2\mathbb{E}\langle \mathbf{e}_{\text{local}}, \mathbf{e}_{\text{global}}\rangle.$$

Cauchy–Schwarz (Steele, 2004) bounds the cross term as

$$\begin{aligned}|\mathbb{E}\langle \mathbf{e}_{\text{local}}, \mathbf{e}_{\text{global}}\rangle| &\leq \mathbb{E}[\|\mathbf{e}_{\text{local}}\|_2 \|\mathbf{e}_{\text{global}}\|_2] \\ &\leq \sqrt{\mathbb{E}\|\mathbf{e}_{\text{local}}\|_2^2}\sqrt{\mathbb{E}\|\mathbf{e}_{\text{global}}\|_2^2}.\end{aligned}$$

Substitution yields

$$\begin{aligned}\mathbb{E}\|\mathbf{e}_{\text{total}}\|_2^2 &\leq \left(\sqrt{\mathbb{E}\|\mathbf{e}_{\text{local}}\|_2^2} + \sqrt{\mathbb{E}\|\mathbf{e}_{\text{global}}\|_2^2}\right)^2, \\ \mathbb{E}\|\mathbf{e}_{\text{total}}\|_2^2 &\geq \left(\sqrt{\mathbb{E}\|\mathbf{e}_{\text{local}}\|_2^2} - \sqrt{\mathbb{E}\|\mathbf{e}_{\text{global}}\|_2^2}\right)^2.\end{aligned} \tag{C.1}$$

Thus, a small expected local error alone does not ensure a small expected total error without control of the global target error. In particular, $\mathbb{E}\|\mathbf{e}_{\text{local}}\|_2^2 = 0$ implies $\mathbb{E}\|\mathbf{e}_{\text{total}}\|_2^2 = \mathbb{E}\|\mathbf{e}_{\text{global}}\|_2^2$: agreement with the EMA target does not by itself establish that the target is accurate.

C.2 Equivalence to Weighted Endpoint Supervision

The student and teacher use the same noisy action a_t and conditioning $z(o, l)$. By Eqs. (3) and (7),

$$f_{\theta_{\text{S}}}(a_t, t) = a_t - t\, v_{\theta_{\text{S}}}(a_t, t), \qquad a^{\text{T}}_0 = a_t - t\, u_{\theta_{\text{T}}}(a_t, t).$$

Subtracting the two endpoints gives

$$f_{\theta_{\text{S}}}(a_t, t) - a^{\text{T}}_0 = -t[v_{\theta_{\text{S}}}(a_t, t) - u_{\theta_{\text{T}}}(a_t, t)]\,. \tag{C.2}$$

Since the teacher target is frozen, stop-gradient does not change its value. Taking squared norms, dividing by t^2, and averaging gives

$$\begin{aligned}\mathcal{L}_{\text{TA}} &= \mathbb{E}_{a_t,t}\left[\|v_{\theta_{\text{S}}}(a_t, t) - u_{\theta_{\text{T}}}(a_t, t)\|_2^2\right] \\ &= \mathbb{E}_{a_t,t}\left[\left\|\frac{a_t - f_{\theta_{\text{S}}}(a_t, t)}{t} - \frac{a_t - a^{\text{T}}_0}{t}\right\|_2^2\right] \\ &= \mathbb{E}_{a_t,t}\left[\frac{\left\|f_{\theta_{\text{S}}}(a_t, t) - a^{\text{T}}_0\right\|_2^2}{t^2}\right], \qquad t > 0.\end{aligned} \tag{C.3}$$

Thus, the velocity loss is exactly a weighted endpoint loss, with $1/t^2$ inside the expectation. The equivalence holds for any teacher solver through the definition of $u_{\theta_{\text{T}}}$.

C.3 Total Endpoint Error Bound

Let $\boldsymbol{\delta} = a_0^{\mathrm{T}} - a_0^{\star}$. Adding this numerical integration error to Eq. (C.2) gives

$$\begin{aligned}\mathbf{e}_{\text{total}} &= \left(f_{\theta_{\mathrm{S}}}(a_t, t) - a_0^{\mathrm{T}}\right) + \boldsymbol{\delta} \\ &= -t[v_{\theta_{\mathrm{S}}}(a_t, t) - u_{\theta_{\mathrm{T}}}(a_t, t)] + \boldsymbol{\delta},\end{aligned}$$

which recovers Eq. (9). Applying a consequence of Cauchy–Schwarz (Steele, 2004)[2], taking expectations, and using $0 < t \leq 1$ yields

$$\begin{aligned}\mathbb{E}\|\mathbf{e}_{\text{total}}\|_2^2 &\leq 2\mathbb{E}\left[t^2 \left\|v_{\theta_{\mathrm{S}}}(a_t, t) - u_{\theta_{\mathrm{T}}}(a_t, t)\right\|_2^2\right] + 2\mathbb{E}\|\boldsymbol{\delta}\|_2^2 \\ &\leq 2\mathcal{L}_{\mathrm{TA}} + 2\mathbb{E}\|\boldsymbol{\delta}\|_2^2.\end{aligned} \tag{C.4}$$

This bounds the combined error $\mathbf{e}_{\text{local}} + \mathbf{e}_{\text{global}}$ and does not require the two components to decrease separately.

Teacher numerical integration error. Following the standard error analysis of the explicit Euler method (Hairer et al., 1993), we estimate the teacher's numerical integration error for a uniform K-step rollout. Let $h = t/K$, $\tau_k = t - kh$, and $\hat{a}_{\tau_k}$ denote the Euler iterates initialized at $\hat{a}_{\tau_0} = a_t$. Assume the teacher velocity is L-Lipschitz in state and $\|d^2 a_\tau^\star / d\tau^2\|_2 \leq M$. For $D_k = \|\hat{a}_{\tau_k} - a_{\tau_k}^\star\|_2$, Taylor expansion gives $D_{k+1} \leq (1 + Lh)D_k + Mh^2/2$, with $D_0 = 0$. Summing this recurrence gives, for $L > 0$,

$$\|\boldsymbol{\delta}\|_2 = D_K \leq \frac{Mh}{2L}\left[(1 + Lh)^K - 1\right] \leq \frac{Me^{Lt}t^2}{2K}.$$

For $L = 0$, the bound is $Mt^2/(2K)$. With uniform L, M, the numerical contribution to Eq. (C.4) is $O(K^{-2})$. Increasing the number of teacher rollout steps K therefore tightens the error bound, providing explicit control over the teacher's numerical integration error.

D Inference Equivalence of Asynchronous Execution

Cross-Expert Wavefront Pipelining (CEWP) changes only the execution schedule, using the same parameters, inputs, attention masks, denoising timestep, and operators as sequential inference. We fix the initial action noise and any future-video noise so that both executions start from identical states (h_0^v, h_0^a). In our implementation, video KV cache entries are published after their required preprocessing, including key normalization and RoPE. The action stream waits on the corresponding event before concatenating video and action KV.

Suppose both executions receive identical states at block i. Equation (10) then gives identical Q/K/V tensors. The event dependency in Eq. (13) ensures that action attention consumes the same completed video KV cache. Applying the same attention and remaining-block operators in Eq. (11) therefore produces identical next-block states for both experts. Induction from $i = 0$ gives, in exact arithmetic,

$$\widehat{h}_i^v = h_i^v, \qquad \widehat{h}_i^a = h_i^a, \qquad i \in \{0, \ldots, N\}. \tag{D.1}$$

where the hat denotes asynchronous execution. The unchanged action output head preserves this equality. CEWP is thus another valid topological ordering of the original computation graph, requiring no architectural modification or additional training, and is independent of TACD.

For sparse conditioning in Algorithm 1, the same argument applies: each event is recorded after the required KV fusion, whose inputs and operator remain unchanged, while non-conditioning action blocks require no video KV. When future-video prediction is retained, the unchanged video output head likewise preserves the video prediction.

E Inference Latency Analysis

We model the execution time of the two experts, excluding common costs such as VAE encoding. Let T_v and T_a denote the standalone latencies of one Video Expert pass and one Action Expert pass,

[2]For real vectors p, q, Cauchy–Schwarz gives $|\langle p, q\rangle| \leq \|p\|_2\|q\|_2$. Together with $2ab \leq a^2 + b^2$, this yields $\|p + q\|_2^2 = \|p\|_2^2 + \|q\|_2^2 + 2\langle p, q\rangle \leq 2\|p\|_2^2 + 2\|q\|_2^2$.

respectively, and S the number of action denoising steps. Since the video KV cache is computed once and reused across steps, sequential execution requires

$$T_{\text{seq}}(S) = T_v + ST_a. \tag{E.1}$$

After TACD, the Action Expert requires only one denoising step ($S = 1$), reducing the sequential latency to

$$T_{\text{seq}}(1) = T_v + T_a. \tag{E.2}$$

CEWP further overlaps the two remaining expert passes through the block-wise schedule in Eq. (13). We model its latency as

$$T_{\text{pipe}}(1) = T_v + T_a - T_{\text{overlap}} + \Delta, \tag{E.3}$$

where T_{overlap} denotes the execution time saved through overlap, and Δ accounts for additional synchronization, scheduling, and resource-contention costs. Under ideal overlap, $T_{\text{overlap}} \approx \min(T_v, T_a)$, giving

$$T_{\text{pipe}}(1) \approx \max(T_v, T_a) + \Delta. \tag{E.4}$$

This is an ideal reference: block dependencies and shared GPU resources limit the overlap attainable in practice. TACD and CEWP thus address complementary sources of latency: TACD reduces the number of action passes, while CEWP overlaps the remaining video and action passes.

F More Experimental Details and Results

F.1 Result sources and aggregation

For results quoted from prior work in Tabs. 1 and 2, we retain the scores and aggregates reported in Light-WAM (Li et al., 2026c), Flash-WAM (Akbari et al., 2026), and Faster-WAM (Zhao et al., 2026). On LIBERO and RoboTwin 2.0, we evaluate Fast-WAM and Faster-WAM using 10 action denoising steps, our implementations of the few-step distillation baselines (DMD, MeanFlow, and consistency distillation), and the RealtimeWAM variants. Unless otherwise specified, RealtimeWAM variants use the action-only LoRA post-training setting described in Sec. 5.1, with the Video Expert kept frozen to maintain the teacher conditioning during distillation. For our evaluations, Overall is the arithmetic mean of the four suite-level success rates on LIBERO and of the Clean and Random success rates on RoboTwin 2.0. On LIBERO-Plus, Overall is the average success rate across perturbation categories, weighted by the number of trials in each category.

F.2 Efficient Kernel Implementation

We apply five operator-level optimizations to reduce kernel launches and intermediate memory traffic. (1) We fuse modulation and residual gating into separate element-wise kernels, preserving BF16 intermediate rounding while keeping LayerNorm independent. (2) We fuse each Q/K RMSNorm into a single Triton kernel, retaining FP32 statistics and BF16 intermediate rounding. (3) We fuse RoPE type conversions and complex rotation into a single Triton kernel while retaining FP64 arithmetic. (4) We replace static all-True attention masks with `None`, preserving masks that exclude attention positions. (5) We pack self-attention Q/K/V and cross-attention K/V projections into single GEMMs, leaving cross-attention Q separate. These optimizations complement CUDA Graph, which reduces CPU submission overhead but does not automatically fuse operators within the graph.

F.3 Detailed LIBERO Results

Tab. F.1 reports success rates on each LIBERO task suite, complementing the overall results in Tab. 1. With one-step action generation, RealtimeWAM* matches the 97.0% overall success rate of 10-step Fast-WAM, while RealtimeWAM† achieves 99.0%, comparable to 10-step Faster-WAM (98.9%). Compared with Faster-WAM, RealtimeWAM† improves Goal success by 1.2 percentage points (99.6% vs. 98.4%), with modest reductions on Spatial and LIBERO-10, indicating that overall performance is preserved despite variations across task suites.

Table F.1: Detailed success rates (%) on the four LIBERO task suites.

Method	NFE		LIBERO				
	Video	Action	Spatial↑	Object↑	Goal↑	LIBERO-10↑	Overall↑
WAM and VLA Baselines							
π_0 (Black et al., 2024)	–	–	96.8	98.8	95.8	85.2	94.1
$\pi_{0.5}$ (Intelligence et al., 2025)	–	–	98.8	98.2	98.0	92.4	96.9
X-VLA (Zheng et al., 2026)	–	–	98.2	98.6	97.8	97.6	98.1
Motus (Bi et al., 2026)	10	10	96.8	99.8	96.6	97.6	97.7
LingBot-VA (Li et al., 2026a)	20	50	98.5	99.6	97.2	98.5	98.5
Fast-WAM (Yuan et al., 2026)	1	10	97.0	99.4	96.6	94.8	97.0
Faster-WAM (Zhao et al., 2026)	1	10	99.2	99.6	98.4	98.4	98.9
Other Efficient Variants							
Flash-WAM (Akbari et al., 2026)	1	2	97.0	92.8	96.4	98.0	96.1
Flash-WAM (Akbari et al., 2026)	1	1	96.0	92.6	96.0	95.8	95.1
Light-WAM (Li et al., 2026c)	1	1	98.2	99.6	97.8	93.0	97.2
Few-Step Distillation							
Fast-WAM (Yuan et al., 2026)	1	1	94.8	96.8	96.6	91.8	95.0
Mean Flow* (Geng et al., 2025)	1	1	97.4	99.6	96.8	93.8	96.9
Consistency Distillation* (Song et al., 2023)	1	1	96.4	99.4	96.8	95.2	96.9
DMD* (Yin et al., 2024)	1	1	97.0	99.4	95.8	92.2	96.1
RealtimeWAM*	1	1	97.0	99.4	97.2	94.2	97.0
RealtimeWAM†	1	1	98.8	99.8	99.6	97.6	99.0

F.4 Additional Ablation Study

We further investigate the start and end points of the teacher rollout used to construct $\mathcal{L}_{\text{TA}}$. We compare three intervals: $1 \to 0$, $t \to r$, and $t \to 0$, where 1 denotes pure noise, 0 denotes the clean endpoint, and t and $r < t$ denote intermediate noise levels. As shown in Tab. F.2, $t \to 0$ achieves the highest overall success rate of 90.84%, outperforming both the full-interval rollout $1 \to 0$ and the intermediate-interval rollout $t \to r$. We therefore adopt $t \to 0$ as the default teacher rollout interval.

Table F.2: Effect of teacher rollout intervals on RoboTwin 2.0. Results are success rates (%); best results are in bold.

Rollout Interval	Clean↑	Random↑	Overall↑
$1 \to 0$	91.26	**89.90**	90.63
$t \to r$	90.96	89.46	90.23
$t \to 0$	**91.96**	89.72	**90.84**

F.5 Latency on Additional GPUs

Tab. F.3 extends the latency evaluation to RTX 5090 and RTX 4090D. TACD alone yields 4.6–5.0× speedups, while CUDA Graph and CEWP further reduce latency across all four model–GPU configurations. With all optimizations enabled, RealtimeWAM achieves 14.1× and 17.7× speedups on Fast-WAM, and 10.0× and 12.4× on Faster-WAM, for RTX 5090 and RTX 4090D, respectively. The current kernels are optimized primarily for NVIDIA H-series GPUs (*e.g.*, H100), which may explain their smaller gains on RTX 5090 and RTX 4090D. We expect dedicated kernel tuning for these GPUs to yield larger speedups.

F.6 Real-World Evaluation of Long-Horizon Garment Folding

We qualitatively evaluate RealtimeWAM* on a real-world T-shirt-folding task using a bimanual setup comprising two PiPER robotic arms from AgileX Robotics[3], with policy inference running on an NVIDIA RTX 4090 GPU. As shown in Figure F.1, the task involves successive grasping and folding operations that transform an unfolded garment into a compact configuration. It requires temporally coordinated decisions because each fold changes the garment geometry and the grasp lo-

[3] https://global.agilex.ai/products/piper

Table F.3: Inference latency (ms) on additional GPUs. Each entry reports latency followed by the cumulative speedup in parentheses, relative to the unoptimized baseline for the same model and GPU.

TACD	CUDA Graph	CEWP	Efficient Kernels	Fast-WAM		Faster-WAM	
				RTX 5090	RTX 4090D	RTX 5090	RTX 4090D
✗	✗	✗	✗	251.5 (1.0×)	480.8 (1.0×)	212.4 (1.0×)	363.7 (1.0×)
✗	✓	✗	✗	97.8 (2.6×)	106.1 (4.5×)	81.4 (2.6×)	98.8 (3.7×)
✓	✗	✗	✗	54.3 (4.6×)	96.8 (5.0×)	43.2 (4.9×)	79.5 (4.6×)
✓	✓	✗	✗	30.8 (8.2×)	35.3 (13.6×)	28.5 (7.5×)	32.3 (11.2×)
✓	✓	✓	✗	24.3 (10.3×)	31.2 (15.4×)	24.4 (8.7×)	29.0 (12.5×)
✓	✓	✓	✓	17.8 (14.1×)	27.2 (17.7×)	21.2 (10.0×)	29.3 (12.4×)



Figure F.1: **Real-world garment folding with AgileX PiPER.** Four keyframes from the recorded rollout show the initial unfolded T-shirt, the configuration after the first fold, a narrow configuration after further lengthwise folding, and the final folded configuration. The rollout was executed with the one-step RealtimeWAM* model. Consecutive actions proceed without perceptible pauses, demonstrating the smooth, low-latency inference that sustains long-horizon deformable-object manipulation.

cations available for subsequent actions. Early misalignment can therefore affect later folds, making this task a useful setting for examining long-horizon manipulation of deformable objects.

To meet the real-time requirement of this sequential setting, we deploy the RealtimeWAM* model distilled from Fast-WAM, using TACD for one-step action generation and CEWP for block-wise overlap between the Video and Action Experts. In the physical evaluation, consecutive actions proceeded without perceptible pauses, enabling smooth real-time execution even without execution-level asynchrony between policy inference and robot execution, such as Real-Time Chunking (RTC) (Black et al., 2025a). The resulting low-latency inference allows the policy to maintain continuous control throughout the long-horizon folding sequence. In the context of RealtimeWAM*, this task motivates preserving action quality under one-step action generation while reducing inference overhead during sequential execution.