You Only Denoise Once: L_1 Fine-Tuning for One-Step Action Prediction in World Action Models

Haodong Li Shaoteng Liu Wei Huang Tianyu Wang Chongjian Ge Shuai Yang Chenzhi Liu Zanyi Wang
Xi Chen Xiaojuan Qi Yukang Chen Zhe Lin Manmohan Chandraker
UCSD Adobe

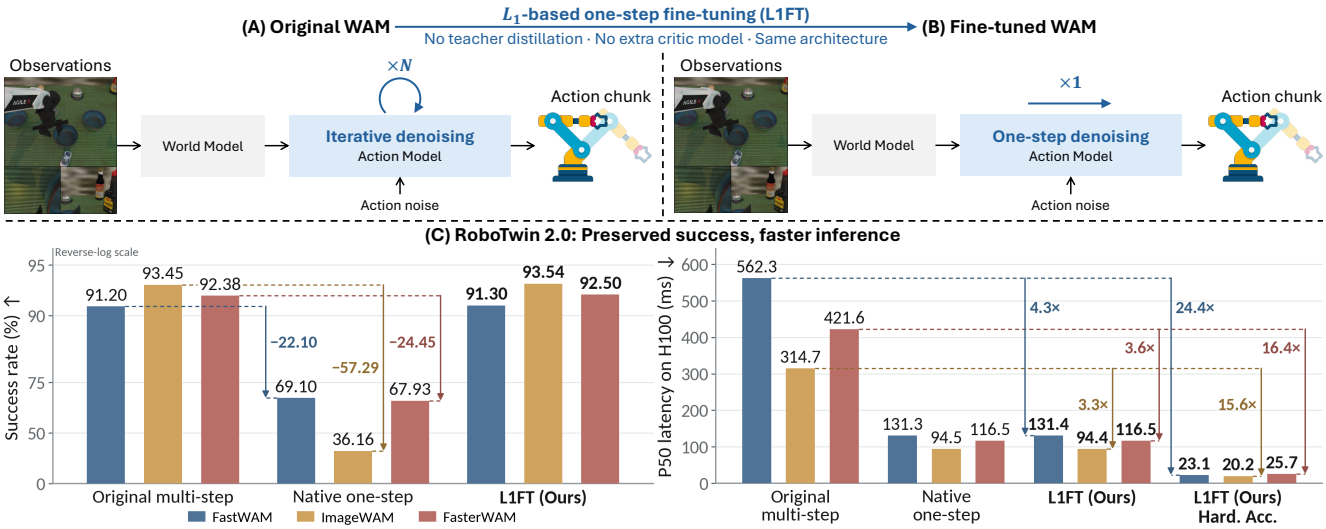


Figure 1. L1FT (L_1-based one-step fine-tuning) turns iterative action denoising into one step while preserving task success. (A) Iterative action denoising remains a major bottleneck to the efficient inference of WAMs, while native one-step denoising (without fine-tuning) can substantially degrade task success. **(B)** L1FT fine-tunes pretrained WAMs on demonstrated actions with an L_1 action loss, enabling one-step denoising with preserved performance. **(C)** On RoboTwin 2.0 [7], L1FT restores the original multi-step success rates of FastWAM [31], ImageWAM [32], and FasterWAM [34] at substantially lower latency, with further speedups from hardware acceleration.

Abstract

Recent advances in world action models (WAMs) have reduced the time spent on future imagination at inference. Yet many of these methods still retain iterative action denoising, which is becoming a major bottleneck of WAMs' inference efficiency. Simply reducing action denoising to one step can substantially degrade task success. We introduce ***L1FT***[1] *(L_1-based one-step fine-tuning), a simple fine-tuning recipe that turns pretrained WAMs from iterative action denoising to one-pass prediction. L1FT directly supervises one-step denoised actions with only an L_1 loss against demonstrated action chunks. It retains the architecture, task conditioning, and non-action training objectives, without relying on teacher distillation or additional critic models. We evaluate L1FT on FastWAM, ImageWAM, and FasterWAM, where multi-step action denoising dominates the inference time. On LIBERO and RoboTwin 2.0, L1FT matches or exceeds all three models' original multi-step success rates with one-pass action prediction. On RoboTwin 2.0, it improves success over native one-step denoising by 22.2, 57.4, and 24.6 percentage points, respectively, and achieves 3.3∼4.3× speedups over the original multi-step policies. With hardware acceleration, the speedup further becomes 15.6∼24.4× (I/O excluded). Real-world dynamic grasping results with FastWAM also support L1FT's ability to replace iterative action denoising with one-pass prediction while preserving task success.*

[1]Can be pronounced as "lift".

1. Introduction

Robots need low-latency action predictions to respond promptly to changes in their environments. World action models use visual world modeling to guide the prediction of the next action chunk from current observations, but predicting future visuals and actions can be costly. Recent advances in WAMs [1, 9, 20, 31, 32, 34] have reduced the time spent on future imagination at inference. FastWAM [31] omits future prediction at inference but retains it during

training. ImageWAM [32] replaces dense video generation with image editing. FasterWAM [34] computes future features once and reuses them during action denoising. However, these models still generate each action chunk through iterative denoising. As visual imagination becomes cheaper at inference, iterative action denoising becomes an increasingly significant source of inference latency.

We humans usually reason over a long horizon (e.g., planning fuel stops before a long drive). Unfamiliar situations (e.g., unexpectedly seeing gas priced over $7.00/gallon) can also require explicit reasoning. Once underway, however, we largely rely on practiced skills for routine steering and adjusting speed on familiar roads, without explicitly reasoning before each movement. Similarly, why not have WAMs predict a short action chunk on trained tasks in a single pass instead of using iterative denoising?

Simply reducing action denoising to one step saves computation but can substantially degrade task success. On RoboTwin 2.0 [7], native one-step inference reduces FastWAM's average success rate from 91.20% to 69.10%, ImageWAM's from 93.45% to 36.16%, and FasterWAM's from 92.38% to 67.93% (Fig. 1 and Tab. 2). The challenge is to retain the original multi-step policy's task performance with one-step action denoising.

Established step-reduction methods for image and video generation learn faster samplers using teacher-generated targets or distributional guidance [23, 29, 30]. These distillation approaches have also been adapted to WAMs to reduce action or visual denoising steps [2, 19, 22, 24, 25]. Action chunks have far fewer dimensions than images or videos. This difference motivates us to reconsider whether the distillation procedures developed for visual generation are necessary for action step reduction. We therefore start with a simple adaptation: directly supervising the pretrained WAM's native one-step denoised actions with L_1 regression. We show that simple supervised fine-tuning suffices to recover the original multi-step task performance.

We introduce L_1-based one-step fine-tuning (**L1FT**), a simple recipe that adapts pretrained WAMs from iterative action denoising to one-pass action prediction while preserving task success. L1FT directly supervises action chunks produced by the native one-step sampler with an L_1 action loss against demonstrated actions. It retains the architecture, task conditioning, and non-action training objectives, requiring no teacher distillation or additional critic models (Tab. 1). Demonstrated actions can vary for similar observations and task conditions. Motivated by prior works on action prediction [14, 33], we use L_1 supervision, whose linear penalty limits the relative influence of large prediction errors compared with squared L_2.

We evaluate L1FT on FastWAM [31], ImageWAM [32], and FasterWAM [34], where multi-step action denoising dominates the inference time, taking up 84%, 71%, and 78%, respectively. Our main quantitative comparison uses LIBERO [17] and RoboTwin 2.0 [7]. As shown in Tabs. 4 and 2, L1FT matches or exceeds the original multi-step success rates of all three pretrained WAMs. It also achieves higher success rates than the compared distillation-based step-reduction methods for general visual generation [29] and WAMs [2, 19, 24]. On RoboTwin 2.0, it improves success over native one-step inference by 22.2, 57.4, and 24.6 percentage points for FastWAM, ImageWAM, and FasterWAM, respectively. Across both benchmarks, reducing action denoising from multiple steps to one yields $3.1\times$ to $4.3\times$ inference speedups, with further gains from hardware acceleration. Real-world dynamic grasping results with FastWAM also support L1FT's effectiveness in replacing iterative action denoising with one-pass prediction while preserving task success. Our summarized contributions are:

- We introduce **L1FT**, a simple fine-tuning recipe that directly supervises a pretrained WAM's native one-step action sampler with an L_1 loss. It requires no teacher distillation, auxiliary critic models, or architectural changes.
- Experiments on three pretrained WAMs across LIBERO and RoboTwin 2.0 show that L1FT preserves task success with substantially reduced inference latency. Real-world dynamic grasping with FastWAM further supports its effectiveness.

Table 1. Conceptual comparison of methods for reducing WAMs' action denoising steps through post-training. No teacher distillation means no model-generated targets or guidance, online or offline (including self-distillation). No extra models means no auxiliary guidance from additional score/critic models. No architecture change means no added modules or conditioning inputs. The first three methods were developed for visual generation.

Method	No teacher distillation	No extra score/critic models	No architecture change
Progressive Distillation [23]	✗	✓	✓
DMD [30]	✗	✗	✓
DMD2 [29]	✗	✗	✓
Consistency Policy [22]	✗	✓	✗
OneDP-S [25]	✗	✗	✓
OneDP-D [25]	✗	✓	✓
SnapFlow [19]	✗	✓	✗
Flash-WAM [2]	✗	✓	✓
1Step-WAM [24]	✗	✓	✗
L1FT (Ours)	✓	✓	✓

2. Related Works

2.1. World action models

World action models use visual world modeling to guide action prediction. Unified World Models [36] and Mo-

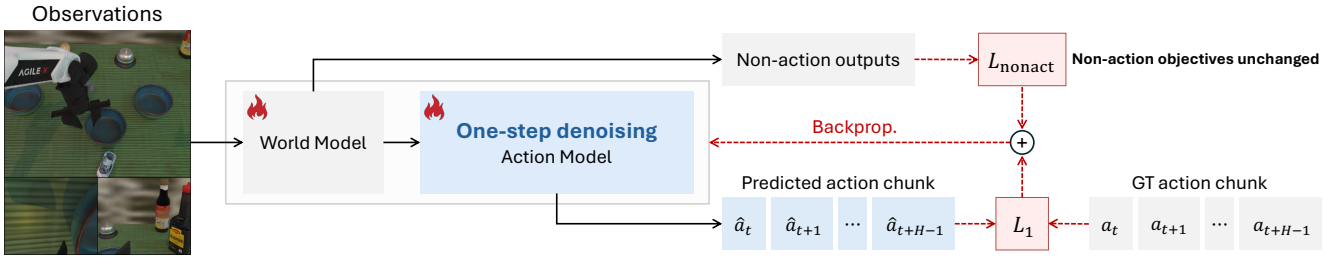


Figure 2. Overview of L1FT. L1FT fine-tunes a pretrained WAM to predict the next action chunk of length H with one-step action denoising. An L_1 loss supervises the native one-step denoised actions against demonstrated actions. L1FT retains the architecture, task conditioning, and non-action objectives L_{nonact}. It requires no teacher distillation or additional critic models. Both world and action models are trainable during fine-tuning. Action noise and task conditions are omitted for clarity.

tus [3] jointly model future videos and actions through iterative denoising. Recent advances in WAMs [1, 6, 9, 20, 31, 32, 34] have reduced the time spent on future imagination at inference. FastWAM [31] omits future prediction at inference but retains it during training. ImageWAM [32] replaces dense video generation with image editing. LaWAM [6] conditions actions on compact future latents. FasterWAM [34] computes future features once and reuses them during action denoising. Yet these WAMs still generate each action chunk through iterative action denoising, which is becoming a significant source of inference latency.

2.2. Step reduction in visual generation

Distillation methods reduce the number of sampling steps in visual generation using model-generated targets or distributional guidance. Progressive Distillation [23] repeatedly replaces two teacher sampling steps with one student step. DMD [30] and DMD2 [29] use teacher scores and an auxiliary model to provide distributional guidance. DMD2 removes the regression loss on teacher-generated targets and adds adversarial supervision from real images.

2.3. Step reduction in WAMs

Recent step reduction methods for WAMs largely follow the distillation approaches discussed in Sec. 2.2. Consistency Policy [22] learns consistency along the pretrained multi-step model's denoising trajectories. OneDP [25] trains a one-step student to match the teacher's action distribution. SnapFlow [19] uses self-distillation with added destination-time conditioning. Flash-WAM [2] applies consistency distillation to both visual and action modalities. 1Step-WAM [24] freezes the pretrained model and uses its multi-step actions to train a velocity calibrator. L1FT instead directly supervises the pretrained WAM's native one-step action output with an L_1 loss against demonstrated actions. It retains the architecture and requires no teacher distillation or additional critic models (Tab. 1).

OFT [14] is also closely related but studies vision-language-action models (VLAs) in a different setting: fine-tuning a pretrained generalist VLA on datasets of new tasks. It replaces autoregressive discrete token prediction in OpenVLA [13] with parallel decoding and a continuous action head. L1FT instead adapts WAMs already pretrained on the target tasks to one-step action prediction while preserving their original multi-step task performance. The training data, testing data, and evaluation protocol remain unchanged.

3. Method

As illustrated in Fig. 2, L1FT fine-tunes a pretrained WAM to predict the next action chunk with one-step action denoising. L1FT reuses the demonstrated actions from the pretraining stage to directly supervise the action chunks produced by its native one-step sampler. We describe the L1FT pipeline in Sec. 3.1, the training losses in Sec. 3.2, and hardware acceleration in Sec. 3.3.

3.1. L1FT: L_1-based one-step fine-tuning

Let $\boldsymbol{c}_t$ denote the current observations and task instruction at the current step. The policy model predicts an action chunk $\widehat{A} = (\hat{\boldsymbol{a}}_t, \ldots, \hat{\boldsymbol{a}}_{t+H-1}) \in \mathbb{R}^{H \times d_a}$ from $\boldsymbol{c}_t$, where H is the prediction horizon and d_a is the action dimension. We focus on pretrained WAMs that predict action chunks through iterative denoising from Gaussian action noise $\boldsymbol{z}_a$. Instead of distilling a student model using distributional guidance or consistency regularization, L1FT directly fine-tunes the native one-step action sampler, initializing the model parameters θ from the pretrained weights:

$$\widehat{A} = \pi_\theta^{(1)}(\boldsymbol{z}_a, \boldsymbol{c}_t), \quad \boldsymbol{z}_a \in \mathbb{R}^{H \times d_a}. \tag{1}$$

Here $\pi_\theta^{(1)}$ shares the same architecture and task conditioning with the pretrained WAM. The superscript (1) denotes one-step action prediction.

During fine-tuning, L1FT directly supervises the output of Eq. (1) with demonstrated actions. At deployment, the same one-step WAM $\pi_\theta^{(1)}$ predicts the next action chunk from the current observation, followed by robot execution.

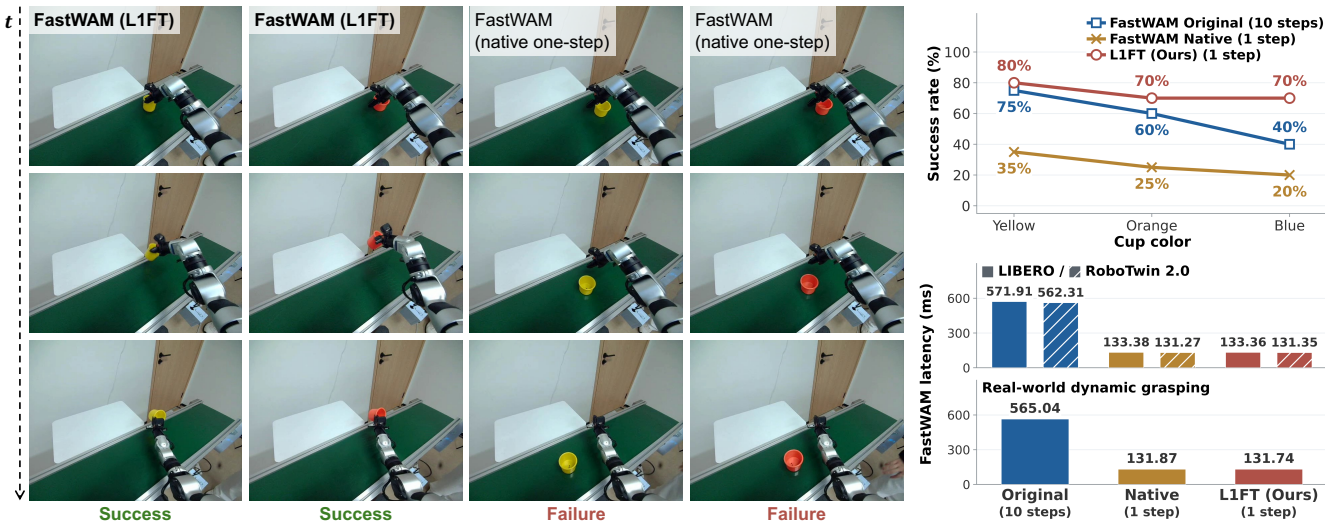


Figure 3. Real-world dynamic grasping with FastWAM [31]. Left: L1FT lifts the moving cup, while native one-step inference fails. Frames at 4.0, 6.0, and 8.0 s are shown from top to bottom. **Right:** real-world success rates by cup color and the inference latency on LIBERO [17], RoboTwin 2.0 [7], and the real-world task. As the cup continues moving during inference, L1FT's lower latency reduces the cup's displacement before execution, helping explain its higher success rate than that of the original ten-step policy. **Raw video recordings are provided in our supplementary material.**

3.2. Training losses

Let $A = (\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+H-1})$ be the demonstrated action chunk. We replace the original action-denoising objective with the mean L_1 error on the one-step denoised actions:

$$L_{\text{act}}(\theta) = \frac{1}{d_a}\frac{1}{H}\sum_{h=0}^{H-1} \|\hat{\boldsymbol{a}}_{t+h} - \boldsymbol{a}_{t+h}\|_1. \tag{2}$$

We average this loss over the minibatch.

Different action sequences can complete the same task from the same observation [8]. Under squared L_2, the gradient magnitude with respect to each predicted action coordinate grows with its absolute error. Under L_1, it is constant for nonzero errors. We therefore use L_1 to reduce the relative influence of large action discrepancies when fitting the demonstrated action chunks [14, 33]. Sec. 4.4 evaluates this choice against squared L_2 supervision.

We jointly fine-tune the world and action models with

$$\mathcal{L}_{\text{L1FT}}(\theta) = \lambda_{\text{act}} L_{\text{act}}(\theta) + L_{\text{nonact}}(\theta). \tag{3}$$

Here, L_{nonact} is the original weighted sum of non-action losses, also averaged over the minibatch. We retain these losses and their weights, together with the weight λ_{act} of the action loss. Among the training losses, only the action objective changes. During fine-tuning, L1FT requires no teacher distillation or auxiliary critic models.

Specifically, for FastWAM [31] and FasterWAM [34], we retain video flow-matching supervision. For ImageWAM [32], we retain image flow-matching supervision. The inherited text encoders and VAEs remain frozen.

3.3. Hardware acceleration

After reducing action denoising to one step, we further lower inference latency with CUDA Graphs and selective compilation. CUDA Graphs capture visual encoding and action prediction at fixed input shapes. Each inference copies current inputs into fixed buffers and replays the graph, reducing CPU dispatch and kernel-launch overhead. Selective compilation fuses smaller elementwise operations into fewer GPU kernels, reducing kernel launches and intermediate memory traffic. We keep the model's original numerical precision before and after hardware acceleration. Given the same inputs, the accelerated model also produces full action chunks that are bitwise identical to those produced before acceleration.

4. Experiments

4.1. Experimental setup

Simulated benchmarks. Our main quantitative evaluation uses LIBERO [17] and RoboTwin 2.0 [7]. On LIBERO, we evaluate all 40 single-arm tasks in the Spatial, Object, Goal, and Long suites, using 50 fixed initial states per task (2,000 episodes). On RoboTwin 2.0, we evaluate all 50 bimanual tasks under Clean and Random conditions, using 100 fixed cases per task per condition (10,000 episodes).

Pretrained WAMs. We evaluate L1FT and other step-reduction baselines on the pretrained checkpoints of FastWAM [31], ImageWAM [32], and FasterWAM [34], where multi-step action denoising dominates inference time. Specifically, iterative action denoising accounts for approximately 84%, 71%, and 78% of inference time, respectively.

Training protocol (simulated benchmarks). We fine-tune these WAMs from their released checkpoints pretrained on the corresponding benchmarks. The training sets contain 1,693 demonstrations for LIBERO and 27,500 for RoboTwin 2.0. We jointly optimize both the world and action models of each pretrained WAM with Eq. (3) using AdamW [18] with BF16 precision. The learning rate is 10^{-5}. The global batch size is 128 on LIBERO and 1,024 on RoboTwin 2.0. The action-loss weight λ_{act} is 1.0. We retain the video flow-matching loss with weight 1.0 for FastWAM and FasterWAM, and the image flow-matching loss with weight 0.5 for ImageWAM. The default training budget is 1.0 epoch. Estimated training time is 7∼8 hours on 8 A100-SXM4-80GB GPUs for LIBERO and 10∼12 hours on 8×8 A100-SXM4-80GB GPUs for RoboTwin 2.0.

Evaluation protocol (simulated benchmarks). We report success rate (SR, %) as the percentage of successful episodes. Average SR is the mean over four LIBERO suites or two RoboTwin 2.0 conditions. We evaluate all L1FT variants and baselines ourselves on A100-SXM4-80GB GPUs, using the same initial states and success criteria. On LIBERO, we allow 400 control steps for Spatial/Object/Goal and 700 for Long, excluding 30 initial settling steps. On RoboTwin 2.0, we use the official task-specific limits of 400∼1,700 control steps. RGB observations comprise external and wrist views on LIBERO, and a head view and two wrist views on RoboTwin 2.0. Under this shared evaluation protocol, we retain each pretrained WAM's official inference settings. FastWAM [31] executes 10 actions between replans on LIBERO and 24 on RoboTwin 2.0. ImageWAM [32] executes 12 and 16. FasterWAM [34] executes 10 and 28.

In the main comparisons (Tabs. 2, 4), we also measure native eager end-to-end (E2E) latency per action chunk on one H100-80GB-HBM3 GPU at batch size 1. We report the median (P50) CUDA-synchronized wall-clock time from CPU observations to CPU actions, including I/O processing and task conditioning. All methods retain their native precision, without the hardware optimizations described in Sec. 4.5.

Baselines (simulated benchmarks). For each pretrained WAM, we compare L1FT with the original multi-step policy and native one-step inference, as well as recent step-reduction methods in both WAMs and visual generation: DMD2 [29], SnapFlow [19], Flash-WAM [2], and 1Step-WAM [24]. For all step-reduction baselines, we follow their official fine-tuning protocols. For a more comprehensive comparison, we also include broader VLA [4, 5, 12, 14, 27, 35] and WAM [3, 6, 10, 15, 16, 26] baselines in Tabs. 2 and 4 using their official inference settings.

Implementation details. The active action dimension d_a is 7 on LIBERO and 14 on RoboTwin 2.0. For L1FT, we sample Gaussian action noise once per pretrained WAM and hold it fixed throughout fine-tuning and inference. This action noise is shared across examples and is not optimized. Sec. 4.4 compares fixed and fresh noise.

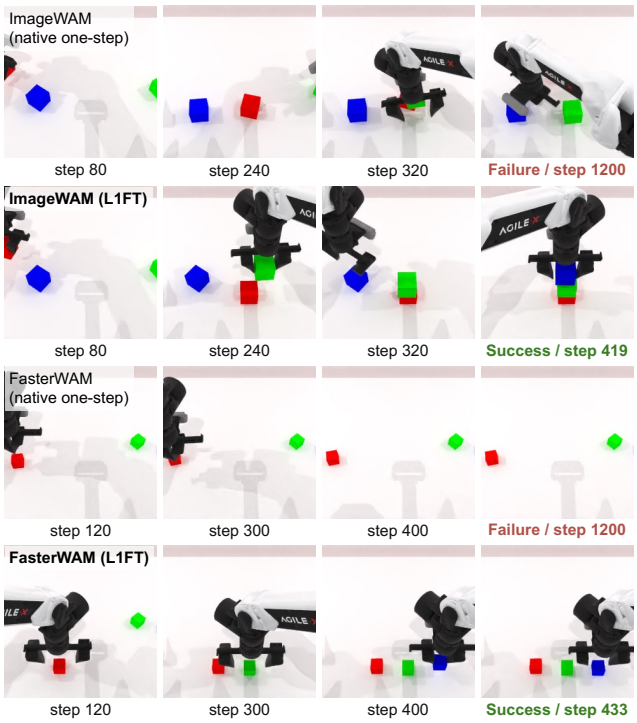


Figure 4. Qualitative comparison on RoboTwin 2.0 [7]. We compare ImageWAM [32] (top) and FasterWAM [34] (bottom) before and after L1FT fine-tuning. Each pair uses the same initial scene and one-step action denoising per chunk. L1FT completes the block stacking task in 419 control steps and block ordering in 433. Both native one-step policies exhaust the 1200-step budget without success.

Real-world benchmark and evaluation protocol. We built a conveyor-grasping platform following LongWAM [11], where a Unitree G1 must grasp and lift a moving cup (Fig. 3). The conveyor belt moves at 3.0 cm/s. We test yellow, orange, and blue cups, with 20 trials per policy and color. Tab. 3 reports successful grasps for the original multi-step, native one-step, and L1FT fine-tuned one-step variants of FastWAM [31]. RGB observations come from the head and right wrist cameras, and the task instruction is "pick up the cup." Each variant predicts chunks of $H = 32$ absolute actions, each with $d_a = 16$ dimensions for 14 arm joints and two grippers.

Real-world training protocol. We fine-tune FastWAM's checkpoint trained on this task using L_1 supervision of native one-step denoised action chunks. We retain its video flow matching loss, with both loss weights set to 1.0. We train only on yellow cups and evaluate the same model on all three colors. The results on orange and blue cups provide evidence of zero-shot generalization, though we do not make broader claims about generalization capabilities.

Table 2. Main quantitative comparison on RoboTwin 2.0 [7]. L1FT matches or exceeds the original ten-step policies' average SRs with one-step action denoising per chunk. Native eager E2E speedups are 3.3× to 4.3× across the three pretrained WAMs. P50 native eager E2E latency is measured per chunk on one H100 at batch size 1, including I/O and task conditioning. Original and native one-step use the same pretrained weights and perform ten-step and one-step action denoising, respectively. Bold and underline mark the best and second-best values overall in each column. Darker and lighter blue mark the best and second-best values among fine-tuned variants within each pretrained WAM.

		Success rate (%) ↑			P50 latency
Base model	Inference policy	Clean	Random	Average	(ms) ↓
π_0 [5]	Original	65.92	58.40	62.16	247.30
X-VLA [35]	Original	72.80	72.84	72.82	143.97
$\pi_{0.5}$ [12]	Original	82.74	76.76	79.75	279.30
LingBot-VLA [27]	Original	87.24	86.18	86.71	750.74
Motus [3]	Original	87.66	85.30	86.48	2869.14
LingBot-VA [16]	Original	90.58	90.72	90.65	4575.06
LaWAM [6]	Original	92.88	89.22	91.05	228.57
LeWAM [10]	Original	91.90	90.66	91.28	179.80
FastWAM [31]	Original	91.76	90.64	91.20	562.31
	Native one-step	70.12	68.08	69.10	131.27
	DMD2 [29]	82.48	83.40	82.94	131.17
	SnapFlow [19]	90.52	86.96	88.74	139.99
	Flash-WAM [2]	90.44	89.80	90.12	131.24
	1Step-WAM [24]	91.10	90.68	90.89	132.34
	L1FT (Ours)	92.24	90.36	91.30	131.35
ImageWAM [32]	Original	93.62	93.28	93.45	314.73
	Native one-step	35.64	36.68	36.16	94.53
	DMD2 [29]	64.52	65.40	64.96	94.59
	SnapFlow [19]	88.00	88.24	88.12	97.96
	Flash-WAM [2]	91.56	91.04	91.30	94.53
	1Step-WAM [24]	92.26	92.62	92.44	95.92
	L1FT (Ours)	**93.64**	**93.44**	**93.54**	**94.36**
FasterWAM [34]	Original	92.86	91.90	92.38	421.60
	Native one-step	67.44	68.42	67.93	116.52
	DMD2 [29]	88.28	86.74	87.51	116.43
	SnapFlow [19]	89.38	89.12	89.25	116.79
	Flash-WAM [2]	92.10	91.46	91.78	116.55
	1Step-WAM [24]	92.38	92.28	92.33	117.33
	L1FT (Ours)	92.64	92.36	92.50	116.52

Table 3. Real-world dynamic grasping. A Unitree G1 grasps cups from a conveyor belt moving at 3.0 cm/s. We report successful grasps out of 20 trials per policy and cup color. Bold and underline mark the best and second-best results per column.

		Color of cup		
Method	Inference policy	Yellow	Orange	Blue
FastWAM	Original	15/20	12/20	8/20
FastWAM	Native 1 step	7/20	5/20	4/20
FastWAM	**L1FT (Ours)**	**16/20**	**14/20**	**14/20**

4.2. Simulated policy performance

Quantitative comparison. As shown in Tabs. 2 and 4, native one-step inference substantially reduces average SR on RoboTwin 2.0 [7], with smaller decreases on LIBERO [17].

L1FT matches or exceeds the original ten-step policies' average SRs across all three pretrained WAMs and both benchmarks. On RoboTwin 2.0, it improves average SR over native one-step inference by 22.2, 57.4, and 24.6 percentage points for FastWAM [31], ImageWAM [32], and FasterWAM [34], respectively. It also achieves the highest average SR among the compared step-reduction methods [2, 19, 24, 29] in all six settings.

Qualitative comparison. In the ImageWAM and FasterWAM examples in Fig. 4, L1FT succeeds within the step budget, whereas native one-step inference exhausts the budget without success.

4.3. Real-world policy performance

Quantitative comparison. Tab. 3 shows that L1FT achieves the highest SR for all three cup colors. It reaches 80%, 70%, and 70% on yellow, orange, and blue cups, respectively, exceeding the original ten-step policy by 5, 10, and 30 percentage points. L1FT also reduces per-chunk inference latency from 565.04 ms to 131.74 ms, a 4.3×

Table 4. Main quantitative comparison on LIBERO [17]. L1FT outperforms the original ten-step policies and all compared step-reduction methods in average SR across the three pretrained WAMs. Timing and notation follow Tab. 2.

Base model	Inference policy	Success rate (%) ↑ Spatial	Object	Goal	Long	Average	P50 latency (ms) ↓
π_0 [5]	Original	96.80	99.20	91.60	83.00	92.65	240.34
GR00T N1.7 [4]	Original	96.80	98.80	97.00	90.80	95.85	146.91
X-VLA [35]	Original	97.00	98.80	96.60	96.20	97.15	136.28
OpenVLA-OFT [14]	Original	97.80	98.60	98.40	95.00	97.45	**90.01**
$\pi_{0.5}$ [12]	Original	98.20	99.00	98.20	96.20	97.90	278.63
LingBot-VA [16]	Original	93.80	99.20	93.80	97.20	96.00	4131.02
Motus [3]	Original	96.80	99.80	96.60	97.60	97.70	2824.22
Cosmos Policy [15]	Original	96.60	99.80	98.00	98.00	98.10	532.42
NowWAM [26]	Original	99.20	**100.00**	**99.80**	95.00	98.50	90.35
LaWAM [6]	Original	98.80	99.60	98.80	98.00	98.80	232.69
FastWAM [31]	Original	97.00	99.40	96.00	93.40	96.45	571.91
	Native one-step	93.60	99.20	97.40	94.20	96.10	133.38
	SnapFlow [19]	96.00	96.80	93.40	78.40	91.15	142.21
	Flash-WAM [2]	97.20	99.20	96.60	95.00	97.00	133.40
	1Step-WAM [24]	98.00	99.00	98.00	93.40	97.10	134.23
	DMD2 [29]	97.40	99.00	97.40	95.00	97.20	133.33
	L1FT (Ours)	96.80	99.20	98.20	95.00	97.30	133.36
ImageWAM [32]	Original	96.20	99.60	98.00	96.20	97.50	311.68
	Native one-step	95.00	99.40	97.60	92.80	96.20	99.56
	SnapFlow [19]	96.00	**100.00**	95.40	92.40	95.95	103.92
	DMD2 [29]	95.00	99.80	95.80	98.20	97.20	99.61
	Flash-WAM [2]	96.20	**100.00**	97.40	97.40	97.75	99.59
	1Step-WAM [24]	96.80	99.80	98.40	96.80	97.95	100.19
	L1FT (Ours)	98.60	99.60	98.00	97.80	98.50	99.04
FasterWAM [34]	Original	**99.60**	99.80	98.00	97.60	98.75	428.86
	Native one-step	99.20	**100.00**	98.20	97.00	98.60	116.68
	SnapFlow [19]	98.60	98.40	95.40	92.00	96.10	117.08
	DMD2 [29]	98.60	**100.00**	99.00	96.60	98.55	116.79
	Flash-WAM [2]	98.40	**100.00**	98.80	98.00	98.80	116.74
	1Step-WAM [24]	98.20	99.80	98.60	**98.60**	98.80	117.62
	L1FT (Ours)	99.00	99.60	98.80	98.20	**98.90**	116.70

speedup (Fig. 3). At 3.0 cm/s, these latencies correspond to approximately 1.70 cm and 0.40 cm of cup motion during inference. Shorter delays reduce the mismatch between the observed scene and the scene at action execution. This helps explain why L1FT outperforms the original ten-step policy.

Qualitative comparison. In Fig. 3, L1FT secures and lifts the cup from the moving belt, while native one-step inference fails to grasp the cups in the shown trials. Both policies use one-step action denoising per chunk.

4.4. Ablation studies

Tabs. 5, 6, and 7 evaluate two changes to one-step diffusion fine-tuning on RoboTwin 2.0 [7]: L_1 action supervision and fixed action noise.

One-step diffusion fine-tuning. We start with fresh Gaussian action noise and the flow-matching MSE at the pure-noise timestep. For one-step action denoising, this is equivalent to squared L_2 supervision of the predicted action chunk. On average, these baselines recover about 90% of the SR lost under native one-step inference, but they still fall short of the original multi-step performance by clear margins.

Table 5. FastWAM [31] ablations on RoboTwin 2.0 [7]. FM denotes flow matching, and the one-step flow-matching MSE loss is equivalent to an L_2 regression loss on the native one-step denoised actions. L_1 improves SR over this baseline, and fixed noise yields a small additional gain. "Original" uses ten action denoising steps, and the other rows use one. SR evaluation and the native eager P50 timing protocols follow Tab. 2. Bold and underline mark the best and second-best SR.

Policy / ablation	Success rate (%) ↑ Clean	Random	Average	P50 latency (ms) ↓
Native one-step	70.12	68.08	69.10	131.27
Fresh noise + L_2 / FM MSE	89.92	87.56	88.74	131.27
Fresh noise + L_1	91.88	89.80	90.84	131.27
Original	91.76	**90.64**	91.20	562.31
Fixed noise + L_1 (L1FT)	**92.24**	90.36	**91.30**	131.35

Table 6. ImageWAM [32] ablations on RoboTwin 2.0 [7]. Notation and timing follow Tab. 5.

	Success rate (%) ↑			P50 latency
Policy / ablation	Clean	Random	Average	(ms) ↓
Native one-step	35.64	36.68	36.16	94.53
Fresh noise + L_2 / FM MSE	90.76	90.20	90.48	94.53
Fresh noise + L_1	93.56	92.00	92.78	94.53
Original	93.62	93.28	93.45	314.73
Fixed noise + L_1 (L1FT)	**93.64**	**93.44**	**93.54**	94.36

Table 7. FasterWAM [34] ablations on RoboTwin 2.0 [7]. Notation and timing follow Tab. 5.

	Success rate (%) ↑			P50 latency
Policy / ablation	Clean	Random	Average	(ms) ↓
Native one-step	67.44	68.42	67.93	116.52
Fresh noise + L_2 / FM MSE	89.74	89.24	89.49	116.52
Fresh noise + L_1	91.98	91.88	91.93	116.52
Original	**92.86**	91.90	92.38	421.60
Fixed noise + L_1 (L1FT)	92.64	**92.36**	**92.50**	116.52

Table 8. Hardware acceleration of L1FT on RoboTwin 2.0 [7]. P50 latency is measured in ms on one H100 at batch size 1. We apply CUDA Graphs and selective compilation, then task-condition precomputation. End-to-end means from CPU observations to CPU actions. FW, IW, and FrW denote FastWAM [31], ImageWAM [32], and FasterWAM [34], respectively. Bold and underline mark the lowest and second-lowest latency per column.

	End-to-end			
Method	Eager	+ Graphs & Compil.	+ Precomp.	No I/O
L1FT (FW)	131.35	**34.53**	**25.82**	23.09
L1FT (IW)	**94.36**	70.10	26.84	**20.18**
L1FT (FrW)	116.52	37.15	28.38	25.72

Table 9. Hardware acceleration of L1FT on LIBERO [17]. Hardware optimizations, timing, and notation follow Tab. 8.

	End-to-end			
Method	Eager	+ Graphs & Compil.	+ Precomp.	No I/O
L1FT (FW)	133.36	**34.80**	**26.14**	**22.33**
L1FT (IW)	**99.04**	85.29	31.18	23.62
L1FT (FrW)	116.70	35.90	27.14	23.69

L_1 action supervision. With fresh noise in both variants, replacing L_2 with L_1 consistently improves average SR by 2.10, 2.30, and 2.44 percentage points for FastWAM [31], ImageWAM [32], and FasterWAM [34], respectively.

Fixed action noise. The L_1 regression objective trains the model to map current observations and task instructions to demonstrated action chunks. Resampling noise varies the predictor's input without changing the target, adding randomness not directly related to its regression target. We therefore fix the Gaussian action noise throughout fine-tuning and inference to simplify the learning of this mapping. This further improves average SR by 0.46, 0.76, and 0.57 percentage points, respectively.

4.5. Inference acceleration

Native eager inference. Tabs. 2 and 4 show that L1FT achieves 3.1× to 4.3× native eager E2E speedups over the original multi-step policies across all pretrained WAMs and both benchmarks. L1FT runs at nearly the same latency as native one-step inference.

Hardware acceleration. Tabs. 8 and 9 report cumulative gains from CUDA Graphs and selective compilation, followed by task-condition precomputation. Together, these optimizations provide a further 3.2× to 5.1× E2E speedup over native eager inference, with P50 latencies of 25.82 to 31.18 ms. Given the same inputs, the accelerated model produces full action chunks that are bitwise identical to those produced before acceleration. Note that ImageWAM [32] gains most from precomputation because its Qwen text encoder [28], which provides the task conditions, accounts for about 45% of the native eager E2E latency (44.78 ms of 99.04 on LIBERO [17] and 41.83 ms of 94.36 on RoboTwin 2.0 [7]). With all optimizations applied, we also measure GPU computation time, excluding CPU I/O processing. This finally yields a 15.6× to 24.4× speedup on RoboTwin 2.0 over the original multi-step policies' E2E latencies.

5. Conclusion

As WAMs spend less time on future imagination at inference, iterative action denoising becomes a major source of latency. Simply reducing action denoising to one step can substantially degrade task success. We introduce L1FT, a simple fine-tuning recipe for adapting pretrained WAMs to one-pass action prediction through direct L_1 supervision of demonstrated action chunks. We evaluate L1FT on FastWAM [31], ImageWAM [32], and FasterWAM [34], where iterative action denoising dominates inference time. On LIBERO [17] and RoboTwin 2.0 [7], L1FT matches or exceeds the original multi-step policies' average success rates with substantially reduced inference latency. Hardware acceleration reduces latency further. Real-world dynamic grasping with FastWAM also supports L1FT's effectiveness.

Limitation and future work. Our work focuses on accelerating short-horizon next-action-chunk prediction on trained tasks. Future work could combine high-level agentic systems for planning and reasoning with a WAM fine-tuned using L1FT for low-latency action prediction, supporting long-horizon and unfamiliar tasks.

6. LLM Usage Statement

We use GPT-6 Astra [21] to edit or refine author-produced content in paper writing, implementation, and experiments. We take full responsibility for all submitted content.

References

[1] Arash Akbari, Arman Akbari, Jingwu Luo, Yuhao Lei, Yi Gao, Weiwei Chen, Xuan Zhang, Zhenman Fang, Geng Yuan, and Yanzhi Wang. Q-wam: 4-bit quantization of world action models with action-subspace protection. *arXiv preprint arXiv:2609.33269*, 2026. 1, 3

[2] 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. 2, 3, 5, 6, 7

[3] 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*, pages 35101–35113, 2026. 3, 5, 6, 7

[4] 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. 5, 7

[5] 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. 5, 6, 7

[6] Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, et al. Lawam: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026. 3, 5, 6, 7

[7] 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. 1, 2, 4, 5, 6, 7, 8

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44 (10-11):1684–1704, 2025. 4

[9] Jiaxin Fang, Ruitong Liu, Zeyu Lai, Xingyu Wang, Mingyuan Jia, Pingyue Sheng, Weikang Wan, Mengdi Xu, and Yuxiang Gao. Bridgewam: Bridging video and action experts via latent bridge queries. *arXiv preprint arXiv:2610.14746*, 2026. 1, 3

[10] Xueji Fang, Boqiang Duan, Hua Wu, Jingdong Wang, and Guo-Jun Qi. Latent evolving world action model. *arXiv preprint arXiv:2609.27455*, 2026. 5, 6

[11] Wei Huang, Bohan Zhang, Chenzhi Liu, Isabella Liu, Shuai Yang, Weian Mao, Luozhou Wang, Yicheng Xiao, Weifeng Lin, Qixin Hu, et al. Long-wam: Scaling the context of world-action models. *arXiv preprint arXiv:2610.10528*, 2026. 5

[12] 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. 5, 6, 7

[13] 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. 3

[14] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. 2, 3, 4, 5, 7

[15] 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. 5, 7

[16] 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*, 2026. 5, 6, 7

[17] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023. 2, 4, 6, 7, 8

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[19] Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. Snapflow: One-step action generation for flow-matching vlas via progressive self-distillation. *arXiv preprint arXiv:2604.05656*, 2026. 2, 3, 5, 6, 7

[20] Liheng Ma, Rui Heng Yang, Zhanguang Zhang, Mateo Clemente, Ziwen Hu, Tongtong Cao, and Yingxue Zhang. Faster-wam: Do world action models need deep action modules? *arXiv preprint arXiv:2608.02365*, 2026. 1, 3

[21] OpenAI. Gpt-6 astra. https://openai.com/index/gpt-6-astra/, 2026. 9

[22] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024. 2, 3

[23] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 2, 3

[24] Rundong Su, Pengtao Chen, Haohan Geng, Sha Zhang, Zhihang Yuan, Yicheng Rong, Chong Yu, Hao Huang, and Tao Chen. 1step-wam: Your world action model but faster using 1-step action generation. *arXiv preprint arXiv:2610.15625*, 2026. 2, 3, 5, 6, 7

[25] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh

Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024. 2, 3

[26] Zanyi Wang, Yuheng Lei, Dengyang Jiang, Ping Luo, Mengdi Wang, Zhixuan Liang, and Shilong Liu. Beyond future prediction: Denoising as generative adaptation for robot control. *arXiv preprint arXiv:2609.28339*, 2026. 5, 7

[27] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026. 5, 6

[28] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 8

[29] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024. 2, 3, 5, 6, 7

[30] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6613–6623. IEEE, 2024. 2, 3

[31] 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. 1, 2, 3, 4, 5, 6, 7, 8

[32] Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. Imagewam: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026. 1, 2, 3, 4, 5, 6, 7, 8

[33] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 2, 4

[34] 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. 1, 2, 3, 4, 5, 6, 7, 8

[35] 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*, pages 60580–60606, 2026. 5, 6, 7

[36] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025. 2