A Simple World Action Model

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

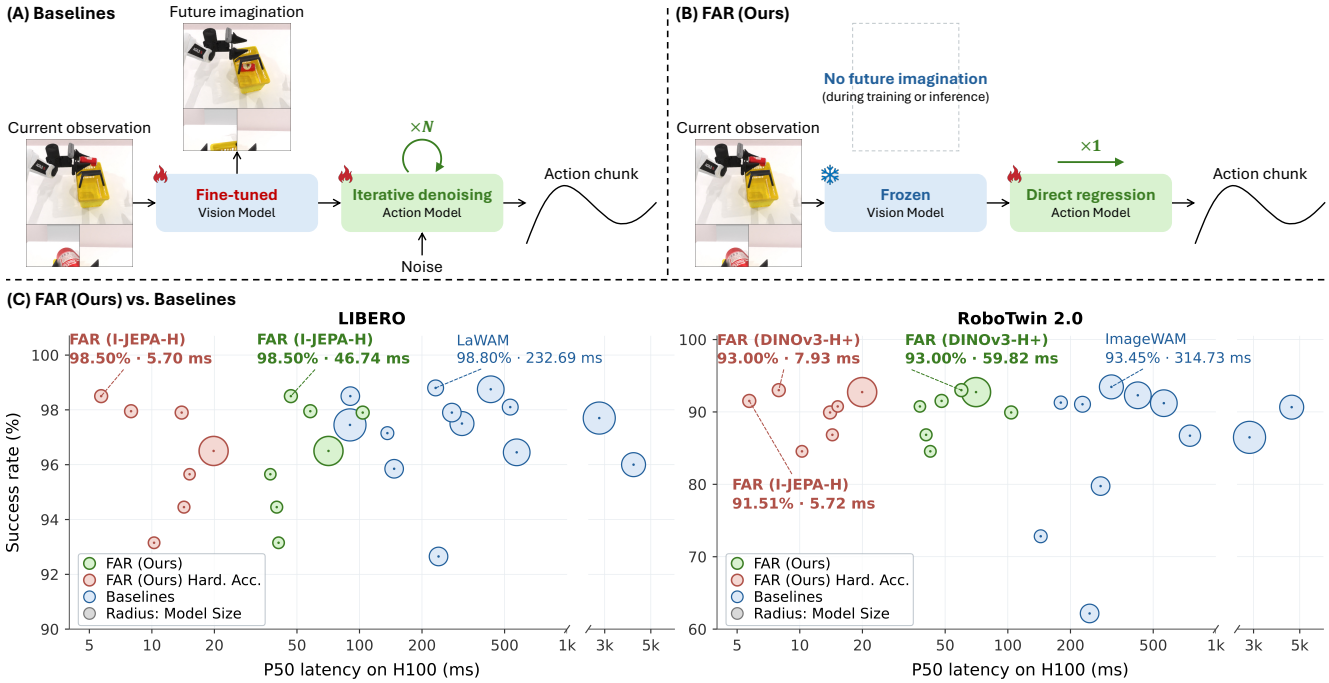


Figure 1. **What does a strong WAM need for next-action-chunk prediction on trained tasks? Good pretrained visual features suffice. (A)** WAMs built on pretrained vision models often use downstream fine-tuning, future imagination, and iterative action denoising. **(B)** FAR learns actions directly from frozen visual features using only an L_1 loss. It predicts an action chunk in one forward pass, without future imagination. **(C)** On LIBERO [22] and RoboTwin 2.0 [11], FAR (green) rivals leading baselines (blue) at lower eager end-to-end (E2E) latency, showing that good pretrained visual features suffice for strong robot policies. Hardware acceleration further brings FAR's fastest configurations (red) below 6 ms on H100, with task conditions precomputed and I/O excluded.

Abstract

World action models (WAMs) use visual world modeling to guide action prediction from current observations. In the setting of next-action-chunk prediction on trained tasks, existing WAMs often use downstream visual fine-tuning, future imagination, and iterative action denoising to turn pretrained vision models into robot policies. But are these mechanisms necessary? We show that good pretrained visual features suffice. We introduce ***FAR*** *(feature-to-action regression), a simple and fast WAM*[1] *recipe that regresses actions directly from frozen visual features using only an L_1 loss. It generates action chunks in one forward pass without future imagination and keeps the vision backbones fixed during training. We implement FAR with seven pretrained visual encoders and evaluate it on LIBERO, RoboTwin 2.0, and real-world conveyor grasping tasks. Our best configurations achieve 98.50% success on LIBERO and 93.00% on RoboTwin 2.0, comparable to leading WAMs. On real-world conveyor grasping tasks, FAR achieves 60% success at 6.0 cm/s, compared with 0% for FastWAM and $\pi_{0.5}$. Thanks to FAR's algorithmic simplicity and hardware acceleration, our fastest configurations achieve median H100 latencies of 5.70 ms on LIBERO and 5.72 ms on RoboTwin 2.0 (I/O excluded). To our knowledge, in this setting, FAR is one of the simplest and fastest (when hardware accelerated) WAM recipes to date*[2]*, with highly competitive performance.*

[1] We use "WAM" broadly to include vision action models such as FAR and NowWAM [34] that do not explicitly predict future observations.

1. Introduction

We humans are always interacting with the physical world using our "policy models". In general, we need reasoning when planning what to do over a long horizon (e.g., deciding which experiments to run before the paper deadline). But in familiar situations, we usually don't reason about what to do in the next moment (e.g., reaching for a coffee mug while waiting for the AI agent). Also, we need reasoning when facing unfamiliar tasks (e.g., making a latte for the first time), while making our usual morning latte can proceed directly from experience, without explicit reasoning.

World action models (WAMs) use visual world modeling to guide the prediction of the next action chunk from current observations. Existing WAMs often use downstream fine-tuning to adapt pretrained visual features to the robotics domain, future imagination to predict possible outcomes, and iterative action denoising to model action distributions [5, 18, 20] (Fig. 1 A). But do WAMs really need these mechanisms for next-action-chunk prediction on familiar tasks? We show that in this setting, good pretrained visual features suffice to learn strong robot policies through direct action regression, without any of these mechanisms.

We propose **FAR** (feature-to-action regression), a simple and fast WAM recipe (Fig. 1 B). Given current RGB observations and a task ID, FAR regresses the next action chunk in one forward pass from multi-layer features of a frozen visual encoder. We train only the action model using only an L_1 loss on demonstrated actions. FAR predicts no future images or latents during policy training or inference. At deployment, the robot executes a prefix of each chunk and replans from new observations.

We instantiate FAR with seven frozen visual encoders pretrained for general visual tasks and evaluate these FAR variants on LIBERO [22] and RoboTwin 2.0 [11]. These encoders come from the JEPA family (I-JEPA-H [3] and V-JEPA2-H [4]), the DINO family (DINOv3-H+ and DINOv3-7B [29]), Wan2.2 I2V [33], MiniMaxH3 FL2VA [1], and FLUX.2 Klein [19]. FAR's best configurations achieve 98.50% success on LIBERO and 93.00% on RoboTwin 2.0, comparable to leading WAMs (Tabs. 2, 3). We further evaluate FAR on a dynamic real-world benchmark where a Unitree G1 grasps a moving cup from a conveyor (Fig. 3). FAR (I-JEPA-H) achieves 60% success at 6.0 cm/s, while both FastWAM and $\pi_{0.5}$ fail in all 20 trials (Tab. 4).

FAR's direct feature-to-action regression also yields low inference latency. Under the same native eager end-to-end (E2E) timing protocol, its configurations with the best success rates are about $5\times$ faster than LaWAM [9] on LIBERO and ImageWAM [38] on RoboTwin 2.0 (Fig. 1 C). We further reduce inference latency with CUDA Graphs, selective compilation, and task condition precomputation, while retaining native precision and keeping full action chunks bitwise identical to native eager outputs. FAR's fastest configurations, both using I-JEPA-H, reach median latencies of 5.70 ms on LIBERO and 5.72 ms on RoboTwin 2.0 with external I/O excluded. These latencies are lower than those of RealtimeWAM [24] under the same timing protocol (Tabs. 7, 8). All latencies are measured on a single H100 GPU at batch size 1. Our summarized contributions are:

Table 1. **Conceptual comparison of WAMs.** The columns indicate the following: frozen vision backbones during policy training; no future image or latent prediction during policy training or inference; and direct regression of action chunks in one forward pass.

Method	Frozen vision	No future imagination	Direct action regression
Motus [5]	✗	✗	✗
Cosmos Policy [18]	✗	✗	✗
LingBot-VA [20]	✗	✗	✗
FastWAM [37]	✗	✗	✗
LaWAM [9]	✗	✗	✗
ImageWAM [38]	✗	✗	✗
FasterWAM [39]	✗	✗	✗
LeWAM [12]	✓	✗	✗
NowWAM [34]	✗	✓	✓
FAR (Ours)	✓	✓	✓

- We ask what a strong WAM actually needs for next-action-chunk prediction on trained tasks and introduce **FAR**, a recipe that regresses action chunks directly from frozen visual features. To our knowledge, in this setting, FAR is one of the simplest and fastest (when hardware accelerated) WAM recipes to date[2], with highly competitive performance.
- We show that good pretrained visual features are sufficient to learn strong robot policies in this setting. On LIBERO and RoboTwin 2.0, FAR delivers success rates comparable to leading WAMs at substantially lower inference latency, especially when hardware accelerated. Real-world conveyor grasping further demonstrates FAR's effectiveness in dynamic manipulation, with substantially higher success rates than FastWAM and $\pi_{0.5}$.

2. Related Work

2.1. General world action models

World action models use visual world modeling to guide the prediction of the next action chunk from current observations. Many existing WAMs [5, 18, 20] follow a common recipe for transferring pretrained visual priors to robot policies. Downstream fine-tuning adapts pretrained generalist visual features to the robotics domain; future imagi-

[2] October 2026.

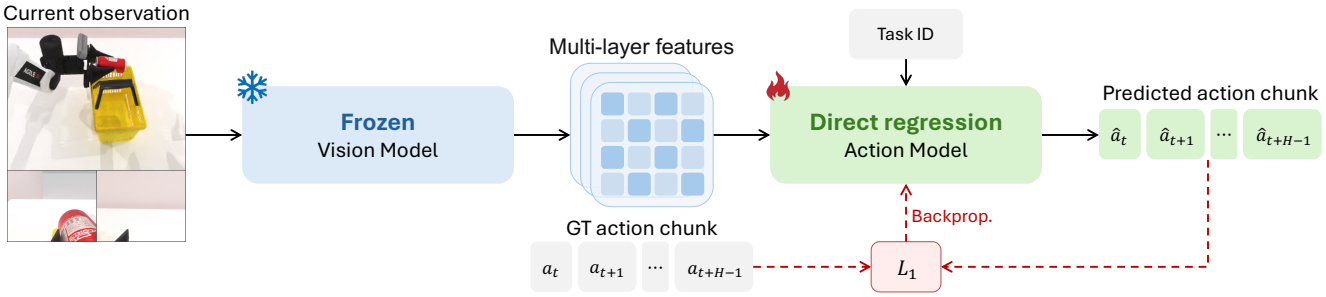


Figure 2. **Overview of FAR.** FAR combines a frozen pretrained vision model with an action model trained from scratch. The vision model extracts multi-layer features from the current observation. The action model's main component is an action transformer, which performs self-attention [32] with Q/K/V from concatenated fused visual and blank action tokens. The task-ID embedding conditions each block via AdaLN [26, 27]. Training minimizes only an L_1 loss between predicted and demonstrated actions.

nation exploits generative visual priors to predict possible outcomes; and iterative action denoising models distributions over multiple valid action sequences.

Some methods condition diffusion policies on generated future images or predictive video features [8, 13, 14, 18, 20, 36]. Others jointly model future observations and actions [5, 28, 42]. These methods have achieved strong performance, but in the setting of next-action-chunk prediction on trained tasks, are these mechanisms necessary for a strong WAM?

2.2. Simplifying WAM recipes

Several recent WAMs explore simpler designs (Tab. 1). ImageWAM [38] replaces video generation with image editing. LaWAM [9] predicts compact visual latents instead of explicit RGB frames. FastWAM [37] omits future prediction at inference and retains it during training. LightWAM [21] directly regresses actions, with future prediction and visual fine-tuning during training. LeWAM [12] freezes its vision backbone and uses training-time future prediction and iterative action denoising. NowWAM [34] removes future prediction and iterative action denoising while fine-tuning its vision backbone through current-observation denoising.

These partial simplifications may suggest that not all mechanisms in Sec. 2.1 are necessary for strong next-action-chunk prediction on trained tasks. This motivates a more fundamental question: What does a strong WAM actually need? To study this question, we introduce FAR, which regresses action chunks directly from frozen visual features (Tab. 1). Our results show that good features suffice to learn strong robot policies in this setting.

2.3. Accelerating WAM inference

Algorithmic approaches accelerate WAMs by reusing computation across denoising steps or action chunks [39, 40] and distilling video and action generation into fewer denoising steps [2, 31]. Hardware optimizations further reduce inference latency through compilation, CUDA Graphs, and optimized kernels [10]. RealtimeWAM [24] combines one-step action distillation and asynchronous video-action execution with hardware optimizations. It is one of the fastest prior WAMs to our knowledge, making it our primary latency baseline for next-action-chunk prediction.

FAR gains algorithmic efficiency from its simple feature-to-action regression recipe, which generates an action chunk in one forward pass. Even without hardware acceleration, FAR is faster than most baselines under the same timing protocol (Tabs. 2, 3). CUDA Graphs, selective compilation, and precomputed task conditions further substantially reduce latency.

3. Method

FAR is a simple and fast WAM recipe for next-action-chunk prediction on trained tasks, combining a frozen pretrained vision model with an action model trained from scratch (Fig. 2). It directly regresses action chunks from frozen visual features without future imagination during policy training or inference. The following subsections describe FAR's pipeline and architecture (Sec. 3.1), training objective (Sec. 3.2), and inference acceleration (Sec. 3.3).

3.1. FAR: feature-to-action regression

Given the current RGB observation I_t and task ID c, FAR predicts a chunk of H actions:

$$\widehat{A}_t = \pi_\theta(I_t, c) \in \mathbb{R}^{H \times d_a}, \tag{1}$$

where d_a is the action dimension, θ represents all trainable parameters of FAR, i.e., the parameters of its action model ϕ_{act}, and $\widehat{A}_t$ is the action chunk $(\hat{\boldsymbol{a}}_t, \ldots, \hat{\boldsymbol{a}}_{t+H-1})$.

The vision model extracts multi-layer features $F_t = \phi^*_{\text{vis}}(I_t)$ from the current observation I_t, where $*$ means all parameters are frozen. The action model's feature fuser maps the multi-layer features to N visual tokens $Z_t = \psi_{\text{fuse}}(F_t) \in \mathbb{R}^{N \times d_z}$. Its task embedder produces a task embedding $\boldsymbol{e}_c = \psi_{\text{task}}(c) \in \mathbb{R}^{d_c}$ from task ID c.

The action transformer ψ_{tran} is the main component of ϕ_{act}. It comprises several transformer blocks, each with

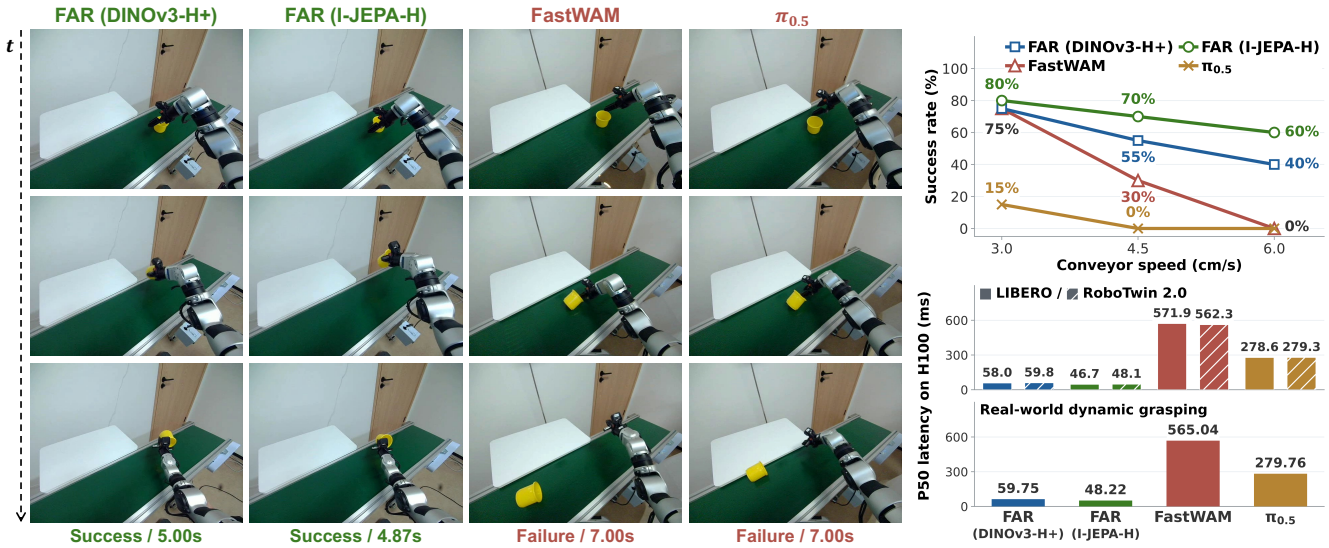


Figure 3. **Qualitative comparison on real-world dynamic grasping. Left:** Both FAR variants lift and retain the moving cup (speed: 4.5 cm/s), whereas FastWAM [37] and $\pi_{0.5}$ [16] leave it toppled on the conveyor. **Right:** FAR's low latency enables its growing advantage as conveyor speed increases. All methods replan from current observations. Longer delays therefore lead to larger displacements before action execution, especially at higher conveyor speeds. **Raw video recordings are provided in our supplementary material.**

self-attention and feedforward network (FFN) layers [32]. It processes the concatenation of Z_t and blank action tokens $\mathbf{0}_{\mathrm{a}} \in \mathbb{R}^{H \times d_z}$ representing zero action inputs:

$$X_t = [\, Z_t; \mathbf{0}_{\mathrm{a}} \,], \quad X'_t = \psi_{\mathrm{tran}}(X_t \mid \boldsymbol{e}_c). \tag{2}$$

In each block, self-attention forms Q/K/V from the concatenation of visual and action tokens. Rotary positional embeddings (RoPE) [30] are applied to Q and K. The task embedding $\boldsymbol{e}_c$ conditions each transformer block through adaptive LayerNorm (AdaLN) [26, 27], which scales and shifts tokens and gates residual updates.

Finally, the action model's prediction head ψ_{head} maps the last H tokens of X'_t to actions:

$$\widehat{A}_t = \psi_{\mathrm{head}}\big([X'_t]_H\big) \in \mathbb{R}^{H \times d_a}, \tag{3}$$

where $[\cdot]_H$ selects the last H tokens. FAR predicts the complete action chunk in one forward pass.

3.2. Training

With demonstrated actions $A_t = (\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+H-1})$, we minimize only the mean L_1 action error:

$$\mathcal{L}_{\mathrm{act}}(\theta) = \frac{1}{d_a} \frac{1}{H} \sum_{h=0}^{H-1} \|\hat{\boldsymbol{a}}_{t+h} - \boldsymbol{a}_{t+h}\|_1. \tag{4}$$

We average this loss over the batch and update only the parameters of the action model ϕ_{act}, i.e., θ in Eq. 1, while keeping ϕ^*_{vis} frozen.

3.3. Inference acceleration

FAR's simple, single-pass action regression is algorithmically efficient and already faster than many baselines under native eager E2E timing (Tabs. 2, 3 and Fig. 1 C).

CUDA Graphs and selective compilation provide hardware acceleration that further reduces latency substantially. 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 combines small operations into fewer GPU kernels, reducing kernel launches and intermediate memory traffic. During hardware optimization, we retain the native precision and require full action chunks stay bitwise identical to the original native eager outputs.

4. Experiments

4.1. Experimental setup

Simulated benchmarks. Our main quantitative evaluation uses LIBERO [22] and RoboTwin 2.0 [11]. 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).

Training protocol (simulated benchmarks). We train FAR on the training sets of both benchmarks, comprising 1,693 LIBERO demonstrations and 27,500 RoboTwin 2.0 demonstrations. For each configuration, we freeze the vision model and train only the action model from scratch

with AdamW [23], using only an L_1 action loss and maintaining an exponential moving average (EMA) of its weights. We train for 10 epochs by default. The global batch size and peak learning rate are 256 and 3×10^{-5} on LIBERO, and 1,024 and 1×10^{-4} on RoboTwin 2.0. Estimated training time is 2∼4 hours on 8 A100-SXM4-80GB GPUs for LIBERO and 4∼10 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 FAR 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 baseline's official inference settings. FAR executes 10 actions between replans on LIBERO and 24 on RoboTwin 2.0. ImageWAM [38] executes 12 and 16. LaWAM [9] executes 8 and 36. FasterWAM [39] executes 28 on RoboTwin 2.0.

In the main comparisons (Tabs. 2, 3), we also measure native eager 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. 3.3. Model size counts both frozen and trainable parameters in the vision and action models.

Baselines (simulated benchmarks). Across the two simulated benchmarks (Tabs. 2, 3), we compare with nine WAMs: Motus [5], Cosmos Policy [18], LingBot-VA [20], FastWAM [37], FasterWAM [39], LaWAM [9], ImageWAM [38], LeWAM [12], and NowWAM [34]. We also compare with six VLAs: π_0 [7], $\pi_{0.5}$ [16], GR00T N1.7 [6], OpenVLA-OFT [17], X-VLA [41], and LingBot-VLA [35].

For the acceleration experiments, we compare with both FastWAM and FasterWAM variants of RealtimeWAM [24], one of the fastest publicly released WAMs to our knowledge. Both variants match FAR's execution intervals of 10 and 24 actions on LIBERO and RoboTwin 2.0, respectively.

FAR implementation. FAR uses seven visual encoders pretrained for general visual tasks, as listed in Sec. 1. To encode current observations directly without future prediction, we use only the V-JEPA2-H [4] encoder, omitting its predictor, and only the VAE encoders of Wan2.2 I2V [33], MiniMaxH3 FL2VA [1], and FLUX.2 Klein [19]. The input resolution is 224×224 for FAR (I-JEPA-H) and 256×256 for the others. FAR (V-JEPA2-H) receives two identical frames of the current observation.

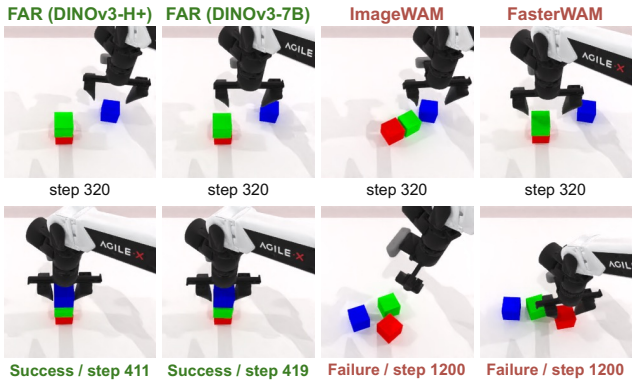


Figure 4. **Qualitative comparisons on RoboTwin 2.0 [11].** Each comparison starts from the same initial scene. The top row shows intermediate states after 320 control steps, and the bottom row shows terminal states. Both FAR variants succeed, while the baselines exhaust the step budget.

Following Sec. 3.1, the feature fuser ψ_{fuse} turns multi-layer features into $N = 256$ visual tokens with $d_z = 1280$. The task embedder ψ_{task} maps task IDs to embeddings with $d_c = 1280$. The action transformer ψ_{tran} has 12 blocks. The prediction head ψ_{head} is a light MLP that maps action tokens to end-effector and gripper commands on LIBERO ($d_a = 7$), or absolute joint and gripper commands on RoboTwin 2.0 ($d_a = 14$). The default prediction horizon is $H = 16$ on LIBERO and $H = 32$ on RoboTwin 2.0. We regress actions in normalized space and denormalize the predicted actions before execution.

Real-world benchmark & evaluation protocol. We built a conveyor-grasping platform following Long-WAM [15], where a Unitree G1 must grasp and lift a moving yellow cup (Fig. 3). FAR takes current RGB images from the head and two wrist cameras, without observation history or robot state. It predicts chunks of $H = 32$ absolute actions, each with $d_a = 16$ dimensions for 14 arm joints and two grippers. We evaluate three conveyor speeds, 3.0, 4.5, and 6.0 cm/s, with 20 trials per method at each speed. We report successful grasps out of 20 for FAR (I-JEPA-H), FAR (DINOv3-H+), FastWAM [37], and $\pi_{0.5}$ [16] (Tab. 4).

Real-world training protocol. We fine-tune all four models from their respective RoboTwin 2.0 checkpoints using 50 demonstrations following their official training protocols. For FAR, we update only the action models with an L_1 loss while keeping the vision backbones frozen.

4.2. Simulated policy performance

Quantitative comparison. FAR achieves success rates comparable to leading WAMs on both LIBERO [22] and RoboTwin 2.0 [11] (Tabs. 2, 3). On LIBERO [22], FAR (I-JEPA-H) reaches an average SR of 98.50%, comparable to LaWAM [9] (98.80%) and FasterWAM [39] (98.75%). On

Table 2. **Main quantitative comparison on LIBERO [22].** FAR (I-JEPA-H) achieves an average SR of 98.50%, comparable to leading WAMs LaWAM [9] (98.80%) and FasterWAM [39] (98.75%). FAR takes 46.74 ms per chunk, about 5× faster than LaWAM (232.69 ms) and 9× faster than FasterWAM (428.86 ms). We evaluate all 40 tasks on A100 with 50 fixed initial states per task (2,000 episodes). Latency is the median (P50) native eager end-to-end (E2E) time on one H100 at batch size 1, without hardware acceleration. We time each fresh action chunk from CPU observations to CPU actions, including I/O processing and task conditioning. Model size counts all vision and action parameters in billions. **Bold** and underlined results mark the best and second best per column.

	Success rate (%) ↑					E2E P50 latency	Model size
Method	Spatial	Object	Goal	Long	Average	H100 (ms) ↓	(B) ↓
π_0 [7]	96.80	99.20	91.60	83.00	92.65	240.34	3
GR00T N1.7 [6]	96.80	98.80	97.00	90.80	95.85	146.91	3
X-VLA [41]	97.00	98.80	96.60	96.20	97.15	136.28	0.9
OpenVLA-OFT [17]	97.80	98.60	98.40	95.00	97.45	90.01	8
$\pi_{0.5}$ [16]	98.20	99.00	98.20	96.20	97.90	278.63	3
LingBot-VA [20]	93.80	99.20	93.80	97.20	96.00	4131.02	5
FastWAM [37]	97.00	99.40	96.00	93.40	96.45	571.91	6
ImageWAM [38]	96.20	99.60	98.00	96.20	97.50	311.68	5
Motus [5]	96.80	99.80	96.60	97.60	97.70	2824.22	8
Cosmos Policy [18]	96.60	99.80	98.00	**98.00**	98.10	532.42	2
NowWAM [34]	99.20	**100.00**	**99.80**	95.00	98.50	90.35	3
FasterWAM [39]	**99.60**	99.80	98.00	97.60	98.75	428.86	6
LaWAM [9]	98.80	99.60	98.80	**98.00**	**98.80**	232.69	2
FAR (Wan2.2 I2V)	96.40	**100.00**	96.80	79.40	93.15	40.77	0.5
FAR (MiniMaxH3 FL2VA)	98.40	99.40	96.80	83.20	94.45	39.95	0.6
FAR (FLUX.2 Klein)	96.20	99.80	96.00	90.60	95.65	**37.22**	**0.4**
FAR (DINOv3-7B)	98.40	**100.00**	98.40	89.20	96.50	70.92	7
FAR (V-JEPA2-H)	98.60	**100.00**	98.80	94.20	97.90	103.67	1
FAR (DINOv3-H+)	99.00	**100.00**	98.20	94.60	97.95	57.97	1
FAR (I-JEPA-H)	99.00	**100.00**	99.00	96.00	98.50	46.74	1

RoboTwin 2.0 [11], FAR (DINOv3-H+) reaches 93.30% on Clean and 92.70% on Random. Its average SR of 93.00% is comparable to ImageWAM [38] (93.45%). Both configurations also outperform the strongest VLA in their respective comparisons: $\pi_{0.5}$ [16] (97.90%) on LIBERO and LingBot-VLA [35] (86.71%) on RoboTwin 2.0. These results support our central finding: good frozen visual features suffice to learn strong policies for next-action-chunk prediction on familiar tasks through direct action regression, without downstream visual fine-tuning, future imagination, or iterative action denoising. In Tab. 2, FAR (I-JEPA-H) achieves 96.00% SR on LIBERO Long, below the 98.00% of LaWAM and Cosmos Policy, despite comparable average SRs. We therefore do not claim that these mechanisms are unnecessary for all tasks or benchmarks, particularly on long-horizon tasks.

Qualitative comparison. Fig. 4 compares selected RoboTwin 2.0 rollouts from the same initial scene. Both FAR variants build the required three-block tower, while ImageWAM [38] and FasterWAM [39] end with separate blocks. FAR completes these successive interactions by replanning from current observations alone, without explicit future imagination or observation history.

Ablation on visual encoders. FAR generally achieves higher SR with JEPA and DINO encoders than with visual encoders from generative models (Tabs. 2, 3). On LIBERO [22], all four JEPA/DINO configurations reach 96.50∼98.50% average SR, above the other three configurations at 93.15∼95.65%. On RoboTwin 2.0 [11], FAR (DINOv3-H+) and FAR (DINOv3-7B) lead among FAR variants at 93.00% and 92.73% average SR, respectively, while the encoders from generative models reach 84.55∼90.75%.

This trend is consistent with their pretraining objectives. The visual encoders of the three generative models compress RGB images into latents and are trained with reconstruction losses [1, 19, 33]. During pretraining, JEPA predicts representations of masked image or video regions [3, 4], while DINO learns representations of whole images and individual patches through teacher-student training [29]. These objectives may make object and spatial information more accessible to the action model, making it easier to learn strong policies from frozen visual features. Good RGB reconstruction does not necessarily yield equally useful features for direct action regression.

4.3. Real-world policy performance

Quantitative comparison. FAR (I-JEPA-H) achieves 80%, 70%, and 60% SR at 3.0, 4.5, and 6.0 cm/s, respectively (Tab. 4), outperforming both FastWAM [37] and $\pi_{0.5}$ [16]

Table 3. **Main quantitative comparison on RoboTwin 2.0 [11].** FAR (DINOv3-H+) achieves an average SR of 93.00%, comparable to the leading WAM, ImageWAM [38] (93.45%). FAR is about 5× faster under native eager E2E timing (59.82 vs. 314.73 ms). We evaluate all 50 tasks across Clean and Random conditions, with 100 fixed cases per task and condition (10,000 episodes). Timing, model size, and notation follow Tab. 2.

	Success rate (%) ↑			E2E P50 latency	Model size
Method	Clean	Random	Average	H100 (ms) ↓	(B) ↓
π_0 [7]	65.92	58.40	62.16	247.30	3
X-VLA [41]	72.80	72.84	72.82	143.97	0.9
$\pi_{0.5}$ [16]	82.74	76.76	79.75	279.30	3
LingBot-VLA [35]	87.24	86.18	86.71	750.74	4
Motus [5]	87.66	85.30	86.48	2869.14	8
LingBot-VA [20]	90.58	90.72	90.65	4575.06	5
LaWAM [9]	92.88	89.22	91.05	228.57	2
FastWAM [37]	91.76	90.64	91.20	562.31	6
LeWAM [12]	91.90	90.66	91.28	179.80	1
FasterWAM [39]	92.86	91.90	92.38	421.60	6
ImageWAM [38]	**93.62**	**93.28**	**93.45**	314.73	5
FAR (Wan2.2 I2V)	87.04	82.06	84.55	42.43	<u>0.5</u>
FAR (MiniMaxH3 FL2VA)	88.40	85.26	86.83	<u>40.49</u>	0.6
FAR (V-JEPA2-H)	90.70	89.14	89.92	103.82	1
FAR (FLUX.2 Klein)	91.88	89.62	90.75	**37.81**	**0.4**
FAR (I-JEPA-H)	92.60	90.42	91.51	48.06	1
FAR (DINOv3-7B)	93.02	92.44	92.73	70.35	7
FAR (DINOv3-H+)	<u>93.30</u>	<u>92.70</u>	<u>93.00</u>	59.82	1

Table 4. **Real-world dynamic grasping.** A Unitree G1 grasps a cup from a moving conveyor belt. We report the number of successful grasps out of 20 per method at each speed. **Bold** and <u>underlined</u> results mark the best and second best per column.

	Conveyor speed (cm/s)		
Method	3.0	4.5	6.0
$\pi_{0.5}$ [16]	3/20	0/20	0/20
FastWAM [37]	<u>15/20</u>	6/20	0/20
FAR (DINOv3-H+)	<u>15/20</u>	<u>11/20</u>	<u>8/20</u>
FAR (I-JEPA-H)	**16/20**	**14/20**	**12/20**

at all speeds. FAR (DINOv3-H+) achieves 75%, 55%, and 40% SR at the same speeds. It outperforms FastWAM at 4.5 and 6.0 cm/s and $\pi_{0.5}$ at all three speeds. At 6.0 cm/s, FAR succeeds in 12 of 20 trials with I-JEPA-H and 8 with DINOv3-H+, while both baselines fail in all 20 trials.

From 3.0 to 6.0 cm/s, SR drops by 25% and 47% for FAR (I-JEPA-H) and FAR (DINOv3-H+), compared with 100% for both FastWAM and $\pi_{0.5}$. FAR's low latency helps explain its growing advantage as conveyor speed increases. All methods replan from current observations, but the cup continues moving during inference. At 6.0 cm/s, the median H100 latencies in Fig. 3 correspond to about 0.29 cm of cup displacement during inference for FAR (I-JEPA-H) and 0.36 cm for FAR (DINOv3-H+). For FastWAM and $\pi_{0.5}$, these distances are 3.39 cm and 1.68 cm. Longer delays therefore cause larger displacements before execution, especially at higher conveyor speeds.

Qualitative comparison. In Fig. 3, both FAR variants successfully grasp, lift, and retain the moving cup at a conveyor speed of 4.5 cm/s. In the illustrated FastWAM and $\pi_{0.5}$ trials, the cups are toppled and left on the belt.

4.4. Inference acceleration

Native eager inference. FAR's simple recipe is algorithmically efficient: it regresses each action chunk in one forward pass, without future imagination or iterative action denoising. Under native eager E2E timing, six of FAR's seven configurations are faster than all baselines in Tabs. 2 and 3. FAR's configurations with the highest SR take 46.74 ms with I-JEPA-H on LIBERO and 59.82 ms with DINOv3-H+ on RoboTwin 2.0. These latencies correspond to 4.98× and 5.26× speedups over the leading baselines on LIBERO and RoboTwin 2.0.

Hardware acceleration. Tabs. 5 and 6 show the cumulative gains from hardware optimization (CUDA Graphs & selective compilation, Sec. 3.3) and task condition precomputation. All accelerated FAR variants in Tabs. 5 and 6 retain native precision and recompute visual features on every request. Their complete action chunks are also bitwise identical to those from native eager inference. Task condition precomputation reuses task embeddings and their AdaLN modulation values within each task. E2E timing spans CPU observations to CPU actions. No I/O excludes external preprocessing and postprocessing. These optimizations reduce FAR (I-JEPA-H)'s E2E latency from 46.74 to 7.81 ms on

Table 5. **Accelerating FAR on LIBERO [22].** Latency is P50 (ms) per action chunk on one H100 at batch size 1. G&C applies CUDA Graphs and selective compilation, with task conditions recomputed per request. Precomp. additionally caches task conditions. No I/O further excludes external preprocessing and postprocessing from E2E timing. Accelerated FAR retains native precision and produces action chunks bitwise identical to those from native eager inference.

	E2E			
Method	Eager	+G&C	+Precomp.	No I/O
FAR (Wan2.2 I2V)	40.77	12.99	12.39	10.25
FAR (MiniMaxH3 FL2VA)	39.95	17.01	16.40	14.28
FAR (FLUX.2 Klein)	**37.22**	17.90	17.42	15.20
FAR (DINOv3-7B)	70.92	22.55	22.17	19.84
FAR (V-JEPA2-H)	103.67	16.79	16.13	13.92
FAR (DINOv3-H+)	57.97	10.62	10.04	7.94
FAR (I-JEPA-H)	46.74	**8.42**	**7.81**	**5.70**

Table 6. **Accelerating FAR on RoboTwin 2.0 [11].** Settings and column definitions follow Tab. 5.

	E2E			
Method	Eager	+G&C	+Precomp.	No I/O
FAR (Wan2.2 I2V)	42.43	13.66	13.02	10.25
FAR (MiniMaxH3 FL2VA)	40.49	17.75	17.09	14.35
FAR (V-JEPA2-H)	103.82	17.44	16.72	13.99
FAR (FLUX.2 Klein)	**37.81**	18.76	18.02	15.22
FAR (I-JEPA-H)	48.06	**9.07**	**8.54**	**5.72**
FAR (DINOv3-7B)	70.35	23.31	22.80	19.95
FAR (DINOv3-H+)	59.82	11.33	10.70	7.93

LIBERO and from 48.06 to 8.54 ms on RoboTwin 2.0.

We further compare FAR with both variants of RealtimeWAM [24] in Tabs. 7 and 8. FAR uses the optimizations described above, while RealtimeWAM uses its official acceleration pipeline. All variants use precomputed task conditions and the same E2E and I/O-excluded timing protocols on one H100-80GB-HBM3 at batch size 1. On LIBERO, FAR (I-JEPA-H) takes 7.81 ms E2E vs. 18.66 ms for RealtimeWAM (FasterWAM), at 98.50% vs. 98.70% SR. With I/O excluded, it takes 5.70 vs. 15.59 ms, a 2.74× speedup. On RoboTwin 2.0, FAR (DINOv3-H+) takes 10.70 ms E2E vs. 19.41 ms for the same RealtimeWAM variant, while achieving 93.00% vs. 91.33% SR. With I/O excluded, it takes 7.93 vs. 16.87 ms, a 2.13× speedup.

5. Conclusion

We ask what a strong WAM actually needs for next-action-chunk prediction on trained tasks. Are downstream visual fine-tuning, future imagination, and iterative action denoising necessary in this setting? FAR shows that good frozen visual features suffice to learn strong policies, through direct action regression with only an L_1 loss. It

Table 7. **FAR vs. RealtimeWAM [24] on LIBERO [22].** SR is the mean over four suites. Latency is P50 (ms) per action chunk on one H100 at batch size 1. All methods use precomputed task conditions. FAR uses the hardware optimizations in Tab. 5, while RealtimeWAM uses its official acceleration pipeline. No I/O further excludes external preprocessing and postprocessing from E2E timing.

		P50 latency (ms) ↓		
Method	SR (%) ↑	E2E	No I/O	Size (B) ↓
RealtimeWAM (FastWAM) [24]	96.20	14.70	11.63	6
RealtimeWAM (FasterWAM) [24]	**98.70**	18.66	15.59	6
FAR (Wan2.2 I2V)	93.15	12.39	10.25	0.5
FAR (MiniMaxH3 FL2VA)	94.45	16.40	14.28	0.6
FAR (FLUX.2 Klein)	95.65	17.42	15.20	**0.4**
FAR (DINOv3-7B)	96.50	22.17	19.84	7
FAR (V-JEPA2-H)	97.90	16.13	13.92	1
FAR (DINOv3-H+)	97.95	10.04	7.94	1
FAR (I-JEPA-H)	98.50	**7.81**	**5.70**	1

Table 8. **FAR vs. RealtimeWAM [24] on RoboTwin 2.0 [11].** SR is the mean over Clean and Random. Settings and column definitions follow Tab. 7.

		P50 latency (ms) ↓		
Method	SR (%) ↑	E2E	No I/O	Size (B) ↓
RealtimeWAM (FastWAM) [24]	90.01	15.13	12.65	6
RealtimeWAM (FasterWAM) [24]	91.33	19.41	16.87	6
FAR (Wan2.2 I2V)	84.55	13.02	10.25	0.5
FAR (MiniMaxH3 FL2VA)	86.83	17.09	14.35	0.6
FAR (V-JEPA2-H)	89.92	16.72	13.99	1
FAR (FLUX.2 Klein)	90.75	18.02	15.22	**0.4**
FAR (I-JEPA-H)	91.51	**8.54**	**5.72**	1
FAR (DINOv3-7B)	92.73	22.80	19.95	7
FAR (DINOv3-H+)	**93.00**	10.70	7.93	1

predicts each action chunk in one forward pass, without future imagination or iterative action denoising, and requires no downstream visual fine-tuning. Across seven pretrained generalist visual encoders, FAR's best configurations achieve 98.50% success on LIBERO [22] and 93.00% on RoboTwin 2.0 [11], comparable to leading WAMs. This direct regression also yields substantially lower native eager E2E latency than most baselines. With hardware acceleration, FAR's fastest configurations achieve median latencies of 5.70 ms on LIBERO and 5.72 ms on RoboTwin 2.0 (H100, I/O excluded). Real-world conveyor grasping further demonstrates FAR's effectiveness in dynamic manipulation.

Limitation and future work. Our experiments use task IDs, evaluate on trained tasks, and focus on short-horizon next-action-chunk prediction. Future work could com-

bine an intelligent but slow agentic "cerebrum" for high-level decisions with FAR as the "cerebellum" for high-frequency execution, supporting long-horizon, generalizable, and natural-language control.

6. LLM Usage Statement

We use GPT-6 Astra [25] to edit or refine author-produced content in paper writing, implementation, and experiments. We take full responsibility for all submitted content.

References

[1] MiniMax AI. Minimax-h3. https://huggingface.co/MiniMaxAI/MiniMax-H3, 2026. 2, 5, 6

[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. 3

[3] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15619–15629. IEEE, 2023. 2, 6

[4] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. 2, 5, 6

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

[6] 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, 6

[7] 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

[8] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *International Conference on Learning Representations*, pages 33431–33452, 2024. 3

[9] 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. 2, 3, 5, 6, 7

[10] Le Chen, Lixin Liu, Jan Schneider, Zeju Qiu, Simon Guist, Bernhard Schölkopf, and Dieter Büchler. Wamjet: A harness for world action model acceleration. *arXiv preprint arXiv:2610.03797*, 2026. 3

[11] 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

[12] Xueji Fang, Boqiang Duan, Hua Wu, Jingdong Wang, and Guo-Jun Qi. Latent evolving world action model. *arXiv preprint arXiv:2609.27455*, 2026. 2, 3, 5, 7

[13] Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024. 3

[14] Siyuan Huang, Liliang Chen, Pengfei Zhou, Shengcong Chen, Yue Liao, Zhengkai Jiang, Yue Hu, Peng Gao, Hongsheng Li, Maoqing Yao, et al. Enerverse: Envisioning embodied future space for robotics manipulation. *Advances in Neural Information Processing Systems*, 38:37693–37720, 2026. 3

[15] 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

[16] 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. 4, 5, 6, 7

[17] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. 5, 6

[18] 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. 2, 3, 5, 6

[19] Black Forest Labs. Flux.2 [klein]. https://bfl.ai/models/flux-2-klein, 2026. 2, 5, 6

[20] 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. 2, 3, 5, 6, 7

[21] 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*, 2026. 3

[22] 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. 1, 2, 4, 5, 6, 8

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

[24] Chengtao Lv, Jinyang Du, Shuyi Feng, Yang Yong, Shiqiao Gu, Shunzi Yang, Ruihao Gong, Shen Ren, Tianwei Zhang, and Wenya Wang. Realtimewam: One-step asynchronous world action models. *arXiv preprint arXiv:2610.06617*, 2026. 2, 3, 5, 8

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

[26] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182. IEEE, 2023. 3, 4

[27] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 3, 4

[28] Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *Advances in neural information processing systems*, 38:95597–95621, 2026. 3

[29] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 2, 6

[30] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 4

[31] 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. 3

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3, 4

[33] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 2, 5, 6

[34] 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. 1, 2, 3, 5, 6

[35] 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, 7

[36] 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. 3

[37] 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. 2, 3, 4, 5, 6, 7

[38] 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. 2, 3, 5, 6, 7

[39] 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. 2, 3, 5, 6, 7

[40] Weisen Zhao, Lam Nguyen, Zhicong Lu, and Yuzhang Shang. C^3ache: Accelerating world action models with cross inference chunk cache. *arXiv preprint arXiv:2606.08962*, 2026. 3

[41] 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

[42] 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. 3