# WAM-ADAPTER: EFFICIENTLY ADAPTING VIDEO-DIT WORLD REPRESENTATIONS TO ROBOT ACTIONS

**Anonymous authors**
Paper under double-blind review

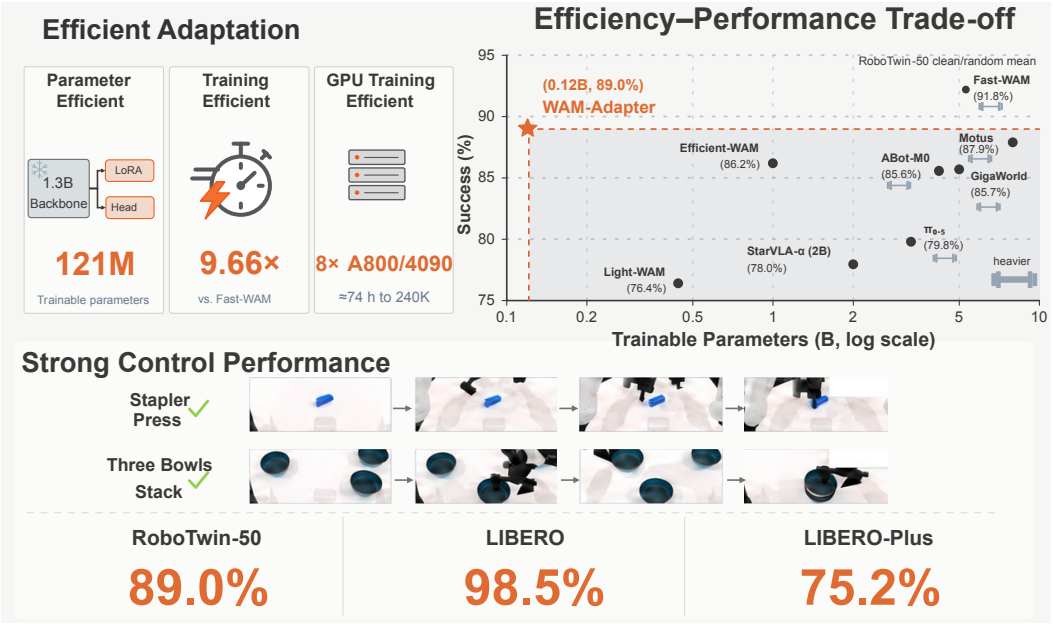


Figure 1: Efficiency–performance summary of WAM-Adapter.

## ABSTRACT

Pretrained Video Diffusion Transformers encode rich spatiotemporal priors from large-scale videos, making them promising backbones for robot control. However, efficiently adapting them into World Action Models remains challenging. Existing approaches reduce video-adaptation or action-prediction costs, but it remains unclear whether representations produced during future-video adaptation can directly support lightweight action prediction. To address this gap, we propose WAM-Adapter, a parameter- and training-efficient framework that freezes the Video-DiT backbone, adapts it with LoRA, and reuses multi-layer representations for action prediction. A lightweight readout organizes multi-layer K/V representations from current-and-future processing into Current, Early-Future, and Late-Future groups, applying group-specific cross-layer fusion for action-chunk prediction. Inference requires a single Video-DiT forward over the current observation and noisy future slots, without future-video generation or iterative action denoising. Without embodied pretraining, WAM-Adapter uses only 121M trainable parameters with a compact Wan-1.3B backbone, achieving 89.0% success across 50 RoboTwin 2.0 tasks, 98.5% on LIBERO, and 75.17% on LIBERO-Plus. To our knowledge, this is the smallest reported trainable-parameter budget for directly trained WAMs without embodied pretraining. Under the same setting, WAM-Adapter improves training throughput by 9.66× while using 98% fewer trainable parameters than Fast-WAM. These results show that Video-DiT representations formed during world-model adaptation can be efficiently reused for strong robot control.

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.

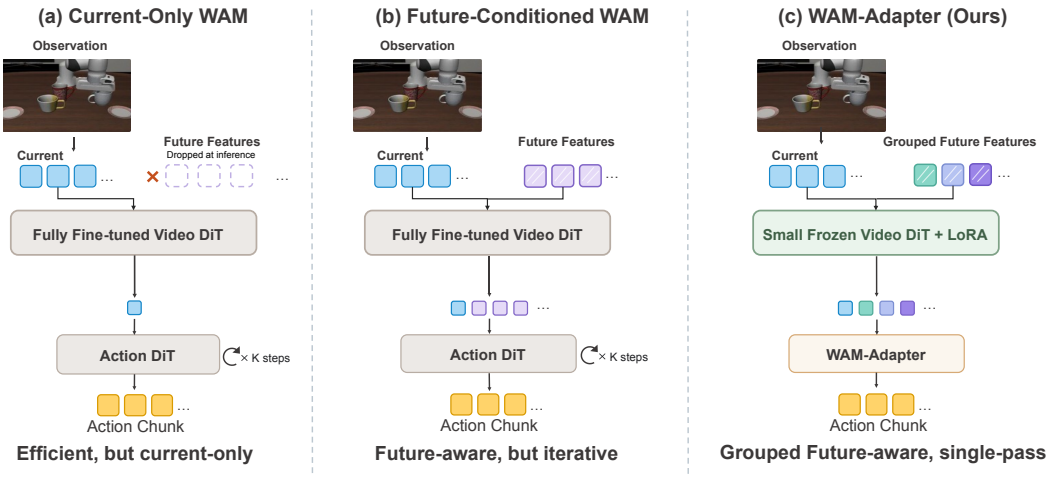


Figure 2: Overview and empirical comparison of WAM-Adapter. It directly predicts action chunks from temporally grouped K/V representations produced by a single Video-DiT forward.

## 1 Introduction

Vision-Language-Action (VLA) models transfer the ability obtained from large-scale vision-language pretraining to robot control and have demonstrated strong multi-task manipulation and generalization capabilities (Kim et al., 2025b). However, many existing VLAs inherit visual representations primarily from static image–text pretraining, leaving physical dynamics and temporal evolution to be learned from limited robot trajectories (Ma et al., 2026b). World Action Models (WAMs) seek to address this limitation by incorporating future-video modeling into policy learning, providing supervision about scene evolution and physical interaction (Yuan et al., 2026; Li et al., 2026c). Pretrained Video Diffusion Transformers are promising backbones for these models because they learn spatiotemporal priors from large-scale video data (Wan Team, 2025; Ma et al., 2026b). Yet adapting these models into robot policies remains expensive, often requiring extensive backbone updates, additional visual computation, or computationally heavy action modules.

To reduce the cost of adapting pretrained Video DiTs for robot control, existing efficient WAMs largely follow two paradigms. The first uses future prediction as a training-time objective but removes future computation during inference, *e.g.,* FastWAM (Yuan et al., 2026), as illustrated in Fig. 2(a). This design improves test-time efficiency, but the policy can no longer directly access future representations when making decisions. The second retains future modeling at inference time, but typically couples it with a separate, iterative process for action generation, *e.g.,* Faster-WAM (Zhao et al., 2026), as shown in Fig. 2(b). These approaches reduce different parts of the WAM pipeline, yet it remains unclear whether the representations formed during future-video adaptation can themselves serve as an efficient interface for action prediction. This motivates us to ask: *whether a pretrained Video DiT can be adapted with limited trainable parameters while directly reusing these representations for lightweight action prediction.*

To answer this question, we introduce **WAM-Adapter**, which makes both Video-DiT adaptation and action readout lightweight by directly reusing the representations produced during future-video adaptation. As illustrated in Figure 2(c), we keep the pretrained Video-DiT backbone frozen and adapt it to robot data using LoRA (Hu et al., 2022), avoiding full backbone fine-tuning. During training, a clean current latent and noisy future latents are processed jointly, producing multi-layer K/V representations across the Video-DiT backbone. Rather than introducing a separate heavy representation path for action prediction, the WAM-Adapter readout directly reuses these K/V representations, preserving them as Current, Early-Future, and Late-Future groups and applying group-specific cross-layer fusion before lightweight direct action decoding. At inference, the future slots are initialized with pure noise and processed together with the current observation in a single Video-DiT forward. This inference path requires neither future-video generation nor an additional action-side Video-DiT forward pass or action denoising.

Experiments demonstrate that WAM-Adapter achieves strong control performance with a compact Video DiT adapted with LoRA. Without embodied pretraining, WAM-Adapter uses a compact 1.3B Video-DiT backbone and only 121M trainable parameters, which is the smallest trainable parameter budget among the directly comparable WAMs in our evaluation. Despite this lightweight adaptation, it achieves 89.0% average success across 50 RoboTwin 2.0 tasks. It further achieves 98.5% on LIBERO and 75.2% on LIBERO-Plus. Beyond simulation, WAM-Adapter also consistently outperforms the lightweight WAM baseline across all three real-world manipulation tasks under the same evaluation protocol, showing that the gains transfer to physical deployment. Under the same setup, WAM-Adapter uses 98.0% fewer trainable parameters and achieves $9.66\times$ the training throughput of Fast-WAM (Yuan et al., 2026). A controlled comparison further shows that LoRA adaptation performs comparably to full fine-tuning. Together, these results demonstrate that WAM-Adapter achieves a strong performance–efficiency trade-off without full backbone fine-tuning.

Our main contributions are summarized as follows:

- We show that strong control performance does not necessarily require a large Video DiT, extensive backbone fine-tuning or heavy action modules.
- We propose **WAM-Adapter**, a parameter-efficient and training-efficient Video-DiT-to-WAM framework. Its readout fuses multi-layer K/V separately for Current, Early-Future, and Late-Future, providing temporally organized context to a lightweight action decoder.
- We extensively evaluate WAM-Adapter on RoboTwin 2.0, LIBERO, LIBERO-Plus, and three real-world manipulation tasks. With only 121M trainable parameters, WAM-Adapter achieves strong multi-task control and out-of-distribution generalization, while consistently outperforming lightweight WAM baseline across all three real-world tasks.

## 2 Related Work

**Vision-Language-Action Models.** VLA models build on large-scale vision-language pretraining to map visual observations and language instructions directly to robot actions. These models have been applied across a wide range of tasks, datasets, and robot embodiments (Brohan et al., 2023; Zitkovich et al., 2023; Open X-Embodiment Collaboration, 2024; Ghosh et al., 2024; Kim et al., 2025b; Liu et al., 2025; Black et al., 2024; P. I. et al., 2025; Bjorck et al., 2025; Pertsch et al., 2025). RT-2 represents robot actions as discrete tokens within a vision-language model, while OpenVLA provides an open pretrained policy that can be fine-tuned for downstream robot tasks (Zitkovich et al., 2023; Kim et al., 2025b). Recent work further improves training and deployment efficiency. OpenVLA-OFT optimizes fine-tuning and action decoding, while TinyVLA and SmolVLA explore compact architectures for efficient robot learning (Kim et al., 2025a; Wen et al., 2024; Shukor et al., 2025). VLA-Adapter introduces learned action queries and lightweight Bridge Attention to extract control-relevant information from vision-language representations with a small policy module (Wang et al., 2026). These methods show that pretrained representations can be adapted to robot control through lightweight interfaces. However, most VLA backbones are pretrained primarily on static image–text data and therefore capture physical dynamics only implicitly through downstream action supervision.

**World Action Models.** Video-based robot policies provide a more explicit temporal learning signal by modeling how visual observations evolve over time. Earlier approaches use predicted videos as plans, visual subgoals, or intermediate representations for action inference (Finn & Levine, 2017; Du et al., 2023; Zhou et al., 2024; Bharadhwaj et al., 2025). UniPi generates language-conditioned video plans before recovering robot actions, whereas Video Prediction Policy conditions inverse dynamics on predictive representations learned by a video diffusion model (Du et al., 2023; Hu et al., 2025). Recent World Action Models (WAMs) couple future visual modeling more directly with robot action learning (Wu et al., 2024; Li et al., 2025; Kim et al., 2026; Ye et al., 2026; Zhou et al., 2026). Systems such as Motus and LingBot-VA jointly learn visual evolution and robot control, while DiT4DiT conditions an Action DiT on intermediate representations obtained during future-video denoising (Bi et al., 2026; Li et al., 2026b; Ma et al., 2026b).

Several recent studies reduce the cost of this video–action pipeline. Fast-WAM retains future-video prediction as training-time supervision but removes video generation during inference (Yuan et al.,

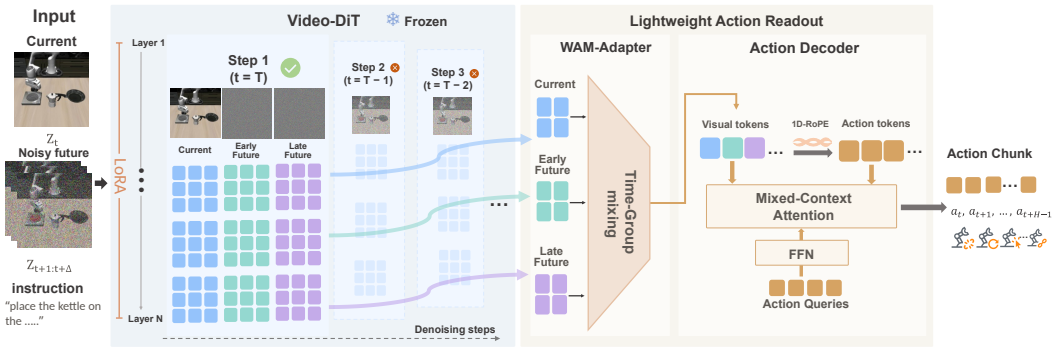


Figure 3: Overview of WAM-Adapter. A frozen pretrained Video DiT is adapted with LoRA and processes the clean current observation together with noisy future latent slots in a single forward pass. WAM-Adapter preserves the resulting Current, Early-Future, and Late-Future representations, applies temporal-group-specific cross-layer readout, and uses a lightweight action decoder to directly predict an action chunk.

2026). Efficient-WAM reduces future-imagination cost through sparse future tokens and asymmetric denoising, whereas Light-WAM combines a compact Video DiT with low-resolution future supervision and an efficient action decoder (Li et al., 2026a;c). ForeWAM and Faster-WAM instead reuse intermediate Video-DiT representations through layer-wise or cross-layer feature fusion (Huang et al., 2026; Zhao et al., 2026; Ma et al., 2026a). Related approaches also explore compact world models, latent prediction, and knowledge transfer to smaller policies (GigaWorld Team et al., 2026; Chen et al., 2026; Jiang et al., 2026). Building on representation reuse, we study a parameter-efficient adaptation setting in which current-and-future K/V representations are fused by temporal group and decoded directly into actions.

## 3 Method

**Problem Formulation.** We consider learning a language-conditioned robot policy from demonstrations. At control step $t$, the policy receives a visual observation $o_t$, a language instruction $l$, and a proprioceptive state $s_t$, and predicts an action chunk $A_t = (a_t, \dots, a_{t+H-1})$ of horizon $H$. Our goal is to adapt a pretrained video diffusion transformer (Video DiT) into an efficient robot policy by reusing its spatiotemporal representations for action prediction.

### 3.1 Shared Video Representations for Action Prediction

WAM-Adapter reuses the K/V representations produced during future-video adaptation through a lightweight WAM-Adapter readout. The readout combines group-specific cross-layer fusion with a lightweight action decoder. LoRA adapts the backbone representations, while the readout learns how to combine and use them for action prediction.

Concretely, we keep the pretrained Video-DiT parameters $\theta$ frozen and adapt the backbone using LoRA parameters $\phi$ (Hu et al., 2022). Let $z_t^0$ denote the clean latent of the current observation and $z_t^f$ the future-video latents. At video flow time $\tau_v$, we construct noisy future latents as

$$z^f_{t,\tau_v} = (1-\tau_v) z_t^f + \tau_v \epsilon^v, \qquad \epsilon^v \sim \mathcal{N}(0, I). \tag{1}$$

The clean current and noisy future latents are then processed together by a single Video forward:

$$\left(\hat{U}^v, \mathcal{C}^v\right) = G_{\theta,\phi}\left([z_t^0; z^f_{t,\tau_v}], C\right), \qquad \mathcal{C}^v = \left\{K^{(\ell)}, V^{(\ell)}\right\}_{\ell=1}^{L}, \tag{2}$$

where $C$ denotes the language and proprioceptive conditioning, $\hat{U}^v$ is the future-video flow prediction, and $\mathcal{C}^v$ is the layer-wise K/V hierarchy produced by the same forward pass.

The action readout reuses these K/V representations without an additional Video-DiT forward pass. It aggregates information across backbone layers while retaining the temporal organization of the tokens.

### 3.2 Temporal Group-Specific Cross-Layer Readout

Recent WAMs have shown that intermediate representations from multiple Video-DiT layers can be reused for action prediction (Zhao et al., 2026; Ma et al., 2026a). However, applying a single cross-layer mixture to the entire video sequence would force current and future representations to share the same depth weighting. We instead preserve their temporal identities and organize the K/V representations into Current, Early-Future, and Late-Future groups. Each group learns its own cross-layer weights, allowing the readout to combine backbone depths differently across temporal groups while preserving the original token structure.

**Temporal grouping.** After temporal compression by the video VAE, the input consists of temporal latent slots $\mathcal{Z} = [\mathcal{Z}_0, \ldots, \mathcal{Z}_{T_z-1}]$. In our default configuration, we assign these slots to three temporal groups:

$$\mathcal{G} = \{\mathcal{G}_{\text{cur}}, \mathcal{G}_{\text{early}}, \mathcal{G}_{\text{late}}\} = \{\mathcal{Z}_0, \mathcal{Z}_1, \mathcal{Z}_2\}, \tag{3}$$

corresponding to the current observation, early future, and late future. This grouping keeps the temporal identities of the video tokens explicit rather than merging representations across time.

**Cross-layer fusion.** Given these temporal groups, we next aggregate the representations of each group across backbone depth. For each Video-DiT layer $\ell$, we extract the canonical K/V representations before 3D RoPE and select the tokens belonging to temporal group $g$, denoted by $K_g^{(\ell)}$ and $V_g^{(\ell)}$. Shared projections map them into the action feature space: $\bar{K}_g^{(\ell)} = K_g^{(\ell)} W_K$ and $\bar{V}_g^{(\ell)} = V_g^{(\ell)} W_V$.

We parameterize the cross-layer mixing weights as $\boldsymbol{\alpha} \in \mathbb{R}^{L_a \times H_a \times G \times L}$, where $L_a$, $H_a$, $G$, and $L$ denote the numbers of action blocks, action attention heads, temporal groups, and Video-DiT layers, respectively. For action block $b$, attention head $h$, and temporal group $g$, we compute the fused representations as

$$\widetilde{K}_{b,h,g} = \sum_{\ell=1}^{L} \alpha_{b,h,g,\ell} \bar{K}_{h,g}^{(\ell)}, \qquad \widetilde{V}_{b,h,g} = \sum_{\ell=1}^{L} \alpha_{b,h,g,\ell} \bar{V}_{h,g}^{(\ell)}. \tag{4}$$

Fusion combines backbone layers rather than tokens, so temporal and spatial token positions remain separate. The same coefficients are applied to keys and values, without softmax normalization or a non-negativity constraint.

We implement this fusion through streaming accumulation: as each Video-DiT layer produces its K/V representations, we accumulate its weighted contribution into the corresponding temporal-group representation. This avoids storing a separate all-layer K/V cache.

**Action-time alignment.** Finally, we align the fused visual keys with the action timeline. The $r$-th video latent slot corresponds to physical action time $p_r = r\Delta$, where $\Delta$ is the temporal stride on the action timeline. We apply layer normalization and action-side 1D RoPE to the fused canonical keys, while leaving the fused values unchanged:

$$\widehat{K}_{b,h,g,s} = R_{1\text{D}}(p_s)\, \text{LN}_b\left(\widetilde{K}_{b,h,g,s}\right), \qquad \widehat{V}_{b,h,g,s} = \widetilde{V}_{b,h,g,s}. \tag{5}$$

### 3.3 Lightweight Direct Action Decoding

Cross-layer fusion combines backbone layers within each temporal group, while action attention weights the resulting tokens for each action query. After action-time alignment, we concatenate the Current, Early-Future, and Late-Future tokens to form the visual context $K_b^v$ and $V_b^v$ for action block $b$. A lightweight action decoder initializes one learned query for each action step and processes all queries jointly using two bidirectional mixed-context Transformer blocks. The resulting action-token features are mapped independently to the complete action chunk by a two-layer MLP head.

**Mixed-context attention.** Let $H_{b-1}^a \in \mathbb{R}^{H \times d_a}$ denote the action-token hidden states entering action block $b$. After pre-normalization, a shared linear projection produces the action-side query,

key, and value representations:

$$Q_b^a,\ K_b^a,\ V_b^a = \mathrm{Split}\left(W_b^{qkv}\,\mathrm{LN}_b^{\mathrm{attn}}(H_{b-1}^a)\right). \tag{6}$$

We apply action-time 1D RoPE to $Q_b^a$ and $K_b^a$ at positions $(1,\ldots,H)$, while $V_b^a$ remains unrotated. In addition to the Current–Early–Late visual K/V, the decoder explicitly reads the proprioceptive state. Specifically, the proprioceptive representation is projected into one state key-value token:

$$h^s = f_s(C_{\mathrm{prop}}), \qquad K^s = W_s^K h^s, \qquad V^s = W_s^V h^s, \tag{7}$$

where $C_{\mathrm{prop}}$ is the encoded proprioceptive context. For action block $b$, the complete mixed context is

$$K_b^c = [K_b^a; K_b^v; K^s], \qquad V_b^c = [V_b^a; V_b^v; V^s], \tag{8}$$

where $[\,;\,]$ denotes concatenation along the token dimension. The action queries attend to all three information sources using a single attention softmax:

$$O_b^a = \mathrm{Softmax}\left(\frac{R_{1\mathrm{D}}(Q_b^a)\,[K_b^c]^\top}{\sqrt{d_h}}\right)V_b^c, \tag{9}$$

where the action-side and visual keys in $K_b^c$ have already been expressed in the action-time 1D RoPE basis, while the state token is appended without positional rotation. This design performs bidirectional self-attention among all action queries while jointly reading the temporally grouped Video-DiT representations and explicit proprioceptive context. Language information is not separately appended to the action decoder; it is already incorporated into the visual K/V through the text-conditioned Video-DiT forward.

**Direct action prediction.** For each action step $i \in \{1,\ldots,H\}$, the decoder initializes an action token by combining a learned query embedding with a projected sinusoidal embedding of its action-time position:

$$H_{0,i}^a = e_i^a + f_{\mathrm{pos}}\left(\mathrm{PE}_{1\mathrm{D}}(i)\right), \qquad i = 1,\ldots,H, \tag{10}$$

where $e_i^a \in \mathbb{R}^{d_a}$ is a learned action query and $f_{\mathrm{pos}}$ is implemented as layer normalization followed by a linear projection and GELU activation. The action queries are jointly processed by $L_a = 2$ action blocks. Each block applies pre-normalized mixed-context attention followed by a pre-normalized feed-forward network:

$$\bar{H}_b^a = H_{b-1}^a + W_b^O O_b^a, \tag{11}$$

$$H_b^a = \bar{H}_b^a + \mathrm{FFN}_b\left(\mathrm{LN}_b^{\mathrm{ffn}}(\bar{H}_b^a)\right), \qquad b = 1,\ldots,L_a. \tag{12}$$

After the final action block, the decoder applies a final layer normalization and an element-wise two-layer prediction MLP to every action token:

$$\hat{a}_{t+i-1} = W_2\,\mathrm{GELU}\left(W_1\,\mathrm{LN}_{\mathrm{out}}\left(H_{L_a,i}^a\right) + b_1\right) + b_2, \qquad i = 1,\ldots,H. \tag{13}$$

Stacking the per-token predictions yields the complete normalized action chunk:

$$\widehat{A}_t = (\hat{a}_t,\ldots,\hat{a}_{t+H-1}) \in \mathbb{R}^{H\times D_a}. \tag{14}$$

The corresponding dataset normalizer maps $\widehat{A}_t$ back to the environment action space at deployment. All $H$ action vectors are predicted simultaneously in one deterministic decoder forward. The decoder is therefore neither autoregressive nor diffusion-based and requires no iterative action denoising.

### 3.4 Joint Training and Single-Pass Inference

**Training objective.** We jointly optimize future-video modeling and direct action regression:

$$\mathcal{L} = \lambda_v \mathcal{L}_{\mathrm{video}} + \lambda_a \mathcal{L}_{\mathrm{action}}, \tag{15}$$

where $\mathcal{L}_{\mathrm{video}}$ is the future-video flow-matching objective and $\mathcal{L}_{\mathrm{action}}$ is an L2 regression loss over the valid action steps. We use equal weighting for the two objectives, setting $\lambda_v = \lambda_a = 1$ in all experiments. Both losses use the same Video-DiT forward pass. Action gradients propagate through the decoder and temporal-group cross-layer readout to the LoRA parameters, while the pretrained backbone weights remain frozen.

**Inference.** At inference time, the current-observation latent remains clean, and future latent slots are initialized with Gaussian noise at $\tau_v = 1$. A single Video-DiT forward pass produces the layer-wise K/V representations, which are fused by the temporal-group-specific readout and passed to the Action Query Decoder to predict the action chunk. The future slots are neither iteratively denoised nor decoded into RGB frames.

# 4 EXPERIMENT

## 4.1 EXPERIMENTAL SETUP

**Benchmarks and data.** We evaluate WAM-Adapter on LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2026), and RoboTwin 2.0 (Chen et al., 2025). For LIBERO, we use the official datasets and report success rates on four suites: Spatial, Object, Goal, and Long. For LIBERO-Plus, we follow the released evaluation protocol of Faster-WAM (Zhao et al., 2026), evaluating the corresponding LIBERO checkpoints zero-shot without additional training. The evaluation covers seven distribution shifts: camera viewpoint, robot initialization, language instruction, lighting, background, sensor noise, and object layout. On RoboTwin 2.0, we train a single policy across 50 tasks using 50 clean and 500 randomized demonstrations per task, and evaluate it under both clean and randomized settings.

**Implementation details.** WAM-Adapter uses Wan2.1-T2V-1.3B (Wan Team, 2025) as the pretrained Video-DiT backbone. Unless otherwise specified, models are optimized with AdamW using a learning rate of $10^{-4}$, weight decay of $10^{-2}$, cosine learning-rate decay, and BF16 precision. For LIBERO, models are trained with a global batch size of 64, while for RoboTwin 2.0, models are trained with a global batch size of 128. Models are trained on NVIDIA A800 GPUs or NVIDIA 4090 GPUS , while evaluation is performed on a single NVIDIA RTX 4090 GPU. Additional architecture and optimization details are provided in the appendix A.

## 4.2 MAIN RESULTS

Table 1 compares policies on LIBERO, RoboTwin 2.0, and LIBERO-Plus. With the same backbone as Light-WAM, WAM-Adapter trains 121M rather than 441M parameters and achieves 98.5%, 89.0%, and 75.17% success, outperforming Light-WAM by 1.3, 12.6, and 19.59 percentage points, respectively. To our knowledge, WAM-Adapter has the smallest trainable-parameter budget among directly trained WAMs without embodied pretraining.

Faster-WAM reaches 99.0% on LIBERO and 92.6% on RoboTwin 2.0, compared with our 98.5% and 89.0%, but uses a 5B backbone with approximately 6.0B trainable parameters. WAM-Adapter achieves the highest LIBERO-Plus success rate of 75.17% among the compared methods.

Table 1: Average success rates (%) on LIBERO, RoboTwin 2.0, and LIBERO-Plus. P.T. denotes embodied pretraining. Backbone denotes the scale of the pretrained foundation backbone. Trainable parameters correspond to the adaptation setting used for the reported LIBERO results.

| Type | Method | P.T. | Backbone | Trainable Params | LIBERO | RoboTwin 2.0 | LIBERO-Plus |
|---|---|---|---|---|---|---|---|
| VLA | OpenVLA (Kim et al., 2025b) | ✓ | 7B | 110.8M | 76.5 | – | – |
| | OpenVLA-OFT (Kim et al., 2025a) | ✓ | 7B | 279M | 97.1 | – | 69.6 |
| | VLA-Adapter (Wang et al., 2026) | × | 0.5B | 197.2M | 97.3 | – | – |
| | $\pi_0$ (Black et al., 2024) | ✓ | 3B | 3.3B | 94.1 | 62.2 | 53.6 |
| | $\pi_{0.5}$ (P. I. et al., 2025) | ✓ | 3B | 3.3B | 96.9 | 79.8 | – |
| WAM | Fast-WAM (Yuan et al., 2026) | × | 5B | 6.0B | 97.6 | 91.8 | 49.1 |
| | LingBot-VA (Li et al., 2026b) | ✓ | 5B | 5.3B | <u>98.5</u> | <u>92.2</u> | – |
| | Faster-WAM (Zhao et al., 2026) | × | 5B | 6.0B | **99.0** | **92.6** | <u>73.6</u> |
| | Light-WAM (Li et al., 2026c) | × | 1.3B | 441.0M | 97.2 | 76.4 | 55.58 |
| | **WAM-Adapter (ours)** | × | 1.3B | **121.0M** | <u>98.5</u> | 89.0 | **75.17** |

## 4.3 EFFICIENCY ANALYSIS

**Training efficiency.** WAM-Adapter achieves 2.29× the training throughput of Light-WAM, where throughput is defined as Throughput $= \frac{\text{number of processed training samples}}{\text{wall-clock training time}}$, while updating 121.0M param-

eters, compared with 441.1M for Light-WAM. Peak GPU memory also decreases from 31.31 to 24.52 GiB. Table 2 reports the training costs of both methods alongside Fast-WAM.

Table 2: Training cost and throughput.

| Method | Backbone | Trainable Params (M) | Peak GPU Mem. (GiB) | Relative Throughput |
|---|---|---|---|---|
| Fast-WAM (Yuan et al., 2026) | 5B | 6,020 | 70.70 | 1.00× |
| Light-WAM (Li et al., 2026c) | 1.3B | 441.1 | 31.31 | 4.22× |
| **WAM-Adapter (ours)** | 1.3B | 121.0 | 24.52 | 9.66× |

**Inference efficiency.** Table 3 compares the inference cost of different WAM designs. Fast-WAM removes future conditioning at inference but still relies on iterative action denoising, whereas Faster-WAM retains future conditioning together with a multi-step action-generation process. In contrast, WAM-Adapter retains future-conditioned representations while directly predicting the action chunk in a single decoder pass. On the same RTX 4090, WAM-Adapter requires 92.05 ms per query and 3.41 GiB of peak GPU memory, compared with 345.36 ms and 12.68 GiB for Fast-WAM, corresponding to 3.75× faster inference and lower peak GPU memory usage.

Table 3: Inference latency and peak GPU memory.

| Method | Backbone | Future Cond. | Action Generation | Latency (ms) | Peak Mem. (GiB) |
|---|---|---|---|---|---|
| Fast-WAM (Yuan et al., 2026) | 5B | × | 10-step denoising | 345.36 | 12.68 |
| Faster-WAM(Zhao et al., 2026) | 5B | ✓ | 10-step denoising | 852.12 | 11.94 |
| **WAM-Adapter (ours)** | 1.3B | ✓ | Direct | 92.05 | 3.41 |

### 4.4 Ablation Studies

We conduct ablations on LIBERO-Long to examine two aspects of WAM-Adapter: the design of the action pathway and the degree of Video-DiT adaptation.

**Action pathway ablations.** The full model reads Current, Early-Future, and Late-Future K/V, applies group-specific cross-layer mixing, and uses all backbone layers. Table 4 removes one design choice at a time. All variants use the same training and evaluation protocol. Without future K/V, the action decoder reads Current K/V only, while the joint future-video training remains in place; success decreases from 97.4% to 88.6%. Sharing mixing weights across temporal groups gives 94.5%. Restricting the readout to layers $\{8, 16, 24\}$ gives 91.3%. Beyond access to future K/V, group-specific fusion and all-layer readout both improve action prediction in these ablations.

| Future K/V | Group mixing | All-layer access | Success (%) |
|---|---|---|---|
| ✗ | ✓ | ✓ | 88.6 |
| ✓ | ✗ | ✓ | 94.5 |
| ✓ | ✓ | ✗ | 91.3 |
| ✓ | ✓ | ✓ | **97.4** |

Table 4: Action-pathway ablations.

**Backbone adaptation.** Figure 4 examines how much Video-DiT adaptation is needed for action prediction. With the backbone frozen and no LoRA, success is 49.4%; adding LoRA substantially improves performance, with the rank-64 setting used by WAM-Adapter reaching 97.4%. Full fine-tuning reaches 98.5%, but updates the entire Video-DiT backbone. Rank 64 is therefore our current efficiency-oriented choice. Additional ablations on the action-decoder depth are provided in Appendix E; two action blocks achieve the highest matched-slice success while remaining substantially lighter than a four-block decoder.

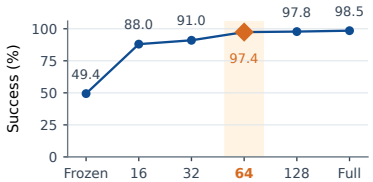


Figure 4: Backbone adaptation.

### 4.5 Qualitative Analysis

We further examine how the learned temporal groups are used across network depth, action horizon, and spatial regions in representative manipulation scenes. All groups peak near the final Video-DiT layers, with similar Early-Future and Late-Future depth profiles (Fig. 5(a)). Their action-step

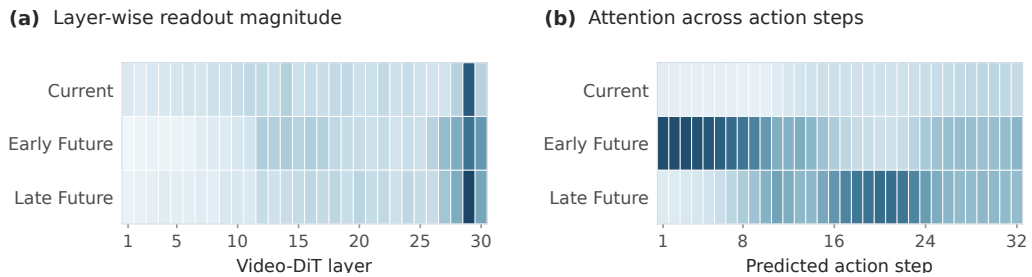


Figure 5: **Temporal-group readout patterns.** (a) Normalized coefficient-weighted key magnitudes across Video-DiT layers. (b) Attention mass across temporal groups and predicted action steps.

attention differs (Fig. 5(b)): Early-Future receives the largest mass near the chunk's beginning, and Late-Future increases toward the middle. Current receives less attention overall, with a moderate increase toward later steps.

Spatial attention also differs across groups (Fig. 6). In mug-to-microwave, the groups attend to overlapping interaction regions with different hotspot locations and extents. In bowl-to-drawer, they focus on more distinct regions, illustrating group-dependent spatial readout.

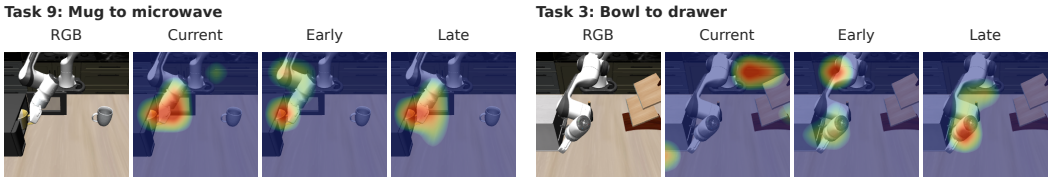


Figure 6: **Spatial attention patterns.** Spatial attention of the Current, Early-Future, and Late-Future groups at the same interaction state within each task.

### 4.6 Real-World Evaluation

We further evaluate WAM-Adapter on an AgileX Piper platform across three real-world manipulation tasks: USB, Keyboard, and Bookshelf. Under the same training data and evaluation protocol, with 20 trials per task using the 150K checkpoint, WAM-Adapter achieves 35%, 25%, and 35% success, compared with 20%, 15%, and 25% for Light-WAM, respectively. Detailed settings are provided in Appendix H.

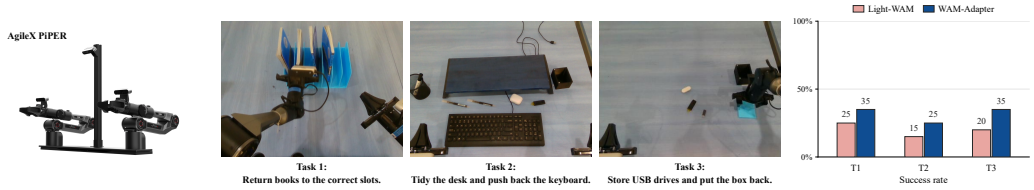


Figure 7: **Real-world evaluation.** Task observations and success rates of Light-WAM (Li et al., 2026c) and WAM-Adapter on three real-world manipulation tasks.

## 5 Conclusion and Discussion

WAM-Adapter combines LoRA adaptation, temporally grouped K/V fusion, and lightweight direct action decoding to efficiently reuse Video-DiT representations for robot control. Experiments on LIBERO, RoboTwin 2.0, and LIBERO-Plus demonstrate a favorable performance–efficiency trade-off. Some larger WAMs still outperform it on RoboTwin 2.0, and performance degrades under distribution shift. Future work will explore adaptation of larger video backbones, more diverse robot data, and robustness-oriented training to address these limitations.

## AI USE STATEMENT

We used generative AI tools solely to assist with language polishing and improving the clarity and organization of the manuscript. All scientific content, analyses, experiments, and conclusions were produced and verified by the authors.

## REFERENCES

Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2Act: Human video generation in novel scenarios enables generalizable robot manipulation. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3936–3951. PMLR, 2025. URL https://proceedings.mlr.press/v305/bharadhwaj25a.html.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Bi_Motus_A_Unified_Latent_Action_World_Model_CVPR_2026_paper.html.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. doi: 10.48550/arXiv.2503.14734.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. $\pi_0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. doi: 10.48550/arXiv.2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025.

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.

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.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026. doi: 10.48550/arXiv.2606.15768.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. doi: 10.48550/arXiv.2506.18088.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 9156–9172, 2023. doi: 10.52202/075280-0403.

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: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/papers/Fei_LIBERO-Plus_A_Progressive_Robustness_Benchmark_for_Visual-Language-Action_Models_CVPR_2026_paper.pdf.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793, 2017. doi: 10.1109/ICRA.2017.7989324. URL https://doi.org/10.1109/ICRA.2017.7989324.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090. URL https://doi.org/10.15607/RSS.2024.XX.090.

GigaWorld Team, Angen Ye, Angyuan Ma, Boyuan Wang, Chaojun Ni, Fangzheng Ye, Guan Huang, Guo Li, Guosheng Zhao, Haodong Yan, Hengtao Li, Jiwen Lu, Kai Wang, Mingming Yu, Qitang Hu, Qiuping Deng, Songling Liu, Xiaoyu Tian, Xiaofeng Wang, Xinyu Zhou, Xiuwei Xu, Xinze Chen, Yang Wang, Yejun Zeng, Yifan Chang, Yun Ye, Zhenyu Wu, Zhanqian Wu, and Zheng Zhu. GigaWorld-Policy-0.5: A faster and stronger WAM empowered by AutoResearch. *arXiv preprint arXiv:2607.13960*, 2026. doi: 10.48550/arXiv.2607.13960.

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. In *International Conference on Learning Representations*, 2022.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25g.html.

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

Yuhua Jiang, Yijun Guo, Hongbing Yang, Guojun Lei, Nuo Chen, Yinuo Zhang, Shaoqiang Yan, Bo Lin, Feifei Gao, and Biqing Qi. CKT-WAM: Parameter-efficient context knowledge transfer between world action models. *arXiv preprint arXiv:2605.06247*, 2026. doi: 10.48550/arXiv.2605.06247.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025a. doi: 10.48550/arXiv.2502.19645.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin

Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025b.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/748becc400a57c0e31cfe6a2e7951467-Abstract-Conference.html.

Jiajun Li, Tiecheng Guo, Yifan Ye, Rongyu Zhang, Xiaowei Chi, Qianpu Sun, Ying Li, Yunfan Lou, Yan Huang, Zhihe Lu, Meng Guo, and Shanghang Zhang. Efficient-wam: A 1b-parameter world-action model with low-cost future imagination. *arXiv preprint arXiv:2606.10040*, 2026a. doi: 10.48550/arXiv.2606.10040.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026b. doi: 10.48550/arXiv.2601.21998.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Proceedings of Robotics: Science and Systems*, Los Angeles, California, USA, June 2025. doi: 10.15607/RSS.2025.XXI.074. URL https://doi.org/10.15607/RSS.2025.XXI.074.

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.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791, 2023. doi: 10.52202/075280-1939.

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*, 2025. URL https://arxiv.org/abs/2410.07864.

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*, 2026a.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026b.

Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL https://doi.org/10.1109/ICRA57147.2024.10611477.

P. I., Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. doi: 10.48550/arXiv.2504.16054.

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. doi: 10.48550/arXiv.2501.09747.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. doi: 10.48550/arXiv.2506.01844.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18638–18646, 2026. doi: 10.1609/aaai.v40i22.38931.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024. doi: 10.48550/arXiv.2409.12514.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2312.13139.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Jim Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. doi: 10.48550/arXiv.2602.15922.

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.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. $\tau_0$-WM: A unified video-action world model for robotic manipulation. *arXiv preprint arXiv:2606.01027*, 2026. doi: 10.48550/arXiv.2606.01027.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. RoboDreamer: Learning compositional world models for robot imagination. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61885–61896. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24f.html.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas,

and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

# A Detailed Architecture

WAM-Adapter inserts rank-64 LoRA adapters into the self-attention and cross-attention Q/K/V/O projections and the two feed-forward projections of every one of the 30 Wan blocks. The pre-trained base weights stay frozen. Canonical K/V are captured before 3D RoPE inside each block. Shared K and V projections map these tensors to the decoder width. The temporal-depth mixing coefficients have shape $\boldsymbol{\alpha} \in \mathbb{R}^{L_a \times H_a \times G \times L}$, where the default model uses $G = 3$ temporal groups and $L = 30$ Video-DiT layers. For each action block $b$, attention head $h$, and temporal group $g \in \{\mathrm{Current}, \mathrm{Early}, \mathrm{Late}\}$, the learned coefficients $\alpha_{b,h,g,:} \in \mathbb{R}^{30}$ accumulate layer-wise K/V representations in a streaming fashion. The same $\alpha_g$ is used for K and V and is not softmax-normalized. This avoids materializing a layer-by-token stack.

The groups are formed using physical video time rather than flattened token indices. In the default configuration, the three temporal latent slots $\mathcal{Z}_0$, $\mathcal{Z}_1$, and $\mathcal{Z}_2$ form the Current, Early-Future, and Late-Future groups, respectively. Their action-time positions are $0$, $\Delta$, and $2\Delta$. A two-block, width-1024, 16-head action-query decoder cross-attends to the resulting K/V. Its feed-forward width is 4096. Learned queries produce a 32-step action chunk; proprioception is projected by a 36,864-parameter linear encoder and supplied to the action readout.

# B Training and Evaluation Details

**Preprocessing.** LIBERO uses two $224 \times 224$ views concatenated consistently across training and evaluation. RoboTwin uses three $240 \times 320$ views and the configured video canvas is $384 \times 320$. Video observations are VAE-encoded, and the cached latents are spatially reduced by a factor of two before the joint pass. The default input contains one current latent slot and two future latent slots, assigned to the Current, Early-Future, and Late-Future groups, respectively. The action horizon is 32 and the action dimension is seven on LIBERO.

**Optimization.** All reported policies use AdamW, learning rate $10^{-4}$, weight decay $0.01$, cosine decay, 1,000 warmup steps, BF16, unit gradient clipping, and seed 42. The video and action objectives are equally weighted with $\lambda_v = \lambda_a = 1$ in all experiments. Gradients from Future tokens to the action path are held at zero for 2K steps, ramped for 3K, and then applied at full scale. Both LIBERO and RoboTwin models are trained on eight NVIDIA A800 GPUs, with global batch sizes of 64 and 128, respectively. Video-DiT and action-decoder gradient checkpointing are disabled in the archived runs.

**Schedules and selection.** The LIBERO runs use 80K-step schedules. LIBERO-Plus evaluates those checkpoints zero-shot. RoboTwin is trained as a 50-task policy and evaluated at 240K. Each LIBERO task uses 50 rollouts; each RoboTwin task uses 50 clean and 50 randomized rollouts.

## C ALGORITHMS

**Algorithm 1** WAM-Adapter Training

*For clarity, action-block and attention-head indices are omitted.*

**Require:** video trajectory $x_{0:T}$, action chunk $A$, language instruction $c$, proprioception $s$
**Require:** Video-DiT $G_{\theta,\phi}$ with frozen base parameters $\theta$ and trainable LoRA parameters $\phi$
**Require:** temporal groups $\mathcal{G} = \{\mathrm{Current}, \mathrm{Early}, \mathrm{Late}\}$

Encode the video trajectory into latent tokens $Z = E_{\mathrm{VAE}}(x_{0:T})$
Keep the current latent clean and perturb the future latents at sampled flow time $\tau_v$
Perform **one joint Video-DiT forward**
    $(\hat{U}^v, \{K^{(\ell)}, V^{(\ell)}\}_{\ell=1}^{L}) \leftarrow G_{\theta,\phi}([z^0; Z^f_{\tau_v}], c, s)$
**for** each temporal group $g \in \mathcal{G}$ **do**
    Fuse all-layer canonical K/V using group-specific coefficients
    $\widetilde{K}_g = \sum_{\ell=1}^{L} \alpha_{g,\ell} \bar{K}_g^{(\ell)}$, $\widetilde{V}_g = \sum_{\ell=1}^{L} \alpha_{g,\ell} \bar{V}_g^{(\ell)}$
    Align the fused keys to the action-time coordinate system
**end for**
Predict the complete action chunk
    $\hat{A} \leftarrow \pi_{\mathrm{act}}(s, \{\widetilde{K}_g, \widetilde{V}_g\}_{g\in\mathcal{G}})$
Compute $\mathcal{L}_{\mathrm{video}}$ and $\mathcal{L}_{\mathrm{action}} = \mathrm{MSE}(\hat{A}, A)$
Update trainable parameters using $\mathcal{L} = \lambda_v \mathcal{L}_{\mathrm{video}} + \lambda_a \mathcal{L}_{\mathrm{action}}$

**Algorithm 2** WAM-Adapter Inference

**Require:** current observation $x_t$, language instruction $c$, proprioception $s$

Encode the current observation as $z_t^0 = E_{\mathrm{VAE}}(x_t)$
Initialize future latent slots with Gaussian noise, $Z^f \sim \mathcal{N}(0, I)$
Perform **one Video-DiT forward**
    $\{K^{(\ell)}, V^{(\ell)}\}_{\ell=1}^{L} \leftarrow G_{\theta,\phi}([z_t^0; Z^f], c, s)$
Apply Current–Early–Late temporal grouping and group-specific cross-layer K/V readout
Align the fused visual keys to action time
Directly predict the complete action chunk $\hat{A}_t = \pi_{\mathrm{act}}(\cdot)$
Execute the configured action prefix and replan

Unlike Light-WAM (Li et al., 2026c), which uses separate backbone computations for future-video adaptation and current-observation action prediction, WAM-Adapter obtains both video supervision and action representations from the same joint Current–Future Video-DiT forward. At inference, it further avoids both iterative future-video generation and iterative action denoising.

## D PARAMETER AND TRAINABLE-COMPONENT BREAKDOWN

Table 5: Parameter breakdown of WAM-Adapter with the Wan2.1-T2V-1.3B backbone. The pre-trained Video-DiT and VAE remain frozen, while LoRA adapters, the video prediction head, the action readout/decoder, and the proprioceptive encoder are trainable. The external frozen text encoder is not included.

| Component | Total | Trainable | Frozen |
|---|---|---|---|
| Frozen Video-DiT base | 1,418,895,360 | 0 | 1,418,895,360 |
| LoRA adapters | 87,490,560 | 87,490,560 | 0 |
| Video prediction head | 101,440 | 101,440 | 0 |
| Action-query decoder/readout | 33,374,791 | 33,374,791 | 0 |
| Proprio encoder | 36,864 | 36,864 | 0 |
| Frozen Wan VAE | 126,892,531 | 0 | 126,892,531 |
| Instantiated model total | 1,666,791,546 | 121,003,655 | 1,545,787,891 |
| Text encoder | external frozen preprocessing; not counted | | |

Table 6: Effect of action-decoder depth on LIBERO-Long.

| Action Blocks | Decoder Params | Δ Params vs. 2 Blocks | Success(%) |
|---|---|---|---|
| 1 | 20.78M | −37.7% | 94.5 |
| **2** | **33.37M** | — | **97.4** |
| 4 | 58.57M | +75.5% | 95.0 |

## E Additional Architecture Ablations

**Action-decoder depth.** We compare one-, two-, and four-block action decoders under the same training and evaluation protocol. The two-block decoder achieves the highest success rate of 97.4%, compared with 94.5% for one block and 95.0% for four blocks. Reducing the decoder to one block decreases the decoder parameter count by 37.7%, but yields lower success. Increasing the depth to four blocks increases decoder parameters by 75.5% without improving performance over the two-block configuration. We therefore use two action blocks as the default setting, which provides the best performance–efficiency trade-off in this comparison.

## F Full RoboTwin-50 Results

Table 7 reports clean/randomized success counts out of 100 percentage points (50 trials per condition). The complete audit contains all 50 tasks; it is split only for readability.

Table 7: RoboTwin tasks 1–25. Entries are Clean/Randomized success (%).

| Task | FastWAM | LightWAM | Ours | Task | FastWAM | LightWAM | Ours |
|---|---|---|---|---|---|---|---|
| turn_switch | 61/59 | 33/39 | 62/50 | beat_block_hammer | 99/97 | 83/80 | 100/100 |
| adjust_bottle | 100/100 | 100/100 | 100/100 | press_stapler | 90/97 | 65/76 | 96/82 |
| stack_bowls_three | 80/81 | 65/72 | 78/72 | open_microwave | 62/45 | 76/59 | 72/80 |
| dump_bin_bigbin | 97/96 | 81/75 | 98/94 | place_cans_plasticbox | 99/96 | 37/68 | 96/98 |
| hanging_mug | 58/62 | 25/17 | 50/58 | move_stapler_pad | 77/64 | 26/34 | 76/74 |
| blocks_ranking_rgb | 100/100 | 96/91 | 100/100 | blocks_ranking_size | 94/98 | 57/54 | 94/82 |
| click_alarmclock | 100/100 | 100/100 | 100/100 | click_bell | 100/100 | 100/100 | 100/100 |
| grab_roller | 100/100 | 100/98 | 100/100 | handover_block | 95/81 | 71/59 | 90/96 |
| handover_mic | 99/100 | 90/94 | 100/100 | lift_pot | 100/100 | 93/93 | 100/98 |
| move_can_pot | 90/88 | 57/74 | 86/86 | move_playingcard_away | 100/100 | 93/92 | 100/100 |
| open_laptop | 98/100 | 91/97 | 98/98 | pick_diverse_bottles | 80/85 | 61/57 | 60/68 |
| pick_dual_bottles | 100/96 | 90/63 | 60/88 | place_a2b_left | 95/93 | 84/83 | 94/90 |
| place_a2b_right | 93/99 | 89/85 | 94/92 | | | | |

Table 8: RoboTwin tasks 26–50 and averages. Entries are Clean/Randomized (%).

| Task | FastWAM | LightWAM | Ours | Task | FastWAM | LightWAM | Ours |
|---|---|---|---|---|---|---|---|
| place_bread_basket | 91/93 | 81/80 | 96/96 | place_bread_skillet | 90/93 | 92/82 | 90/90 |
| place_can_basket | 71/69 | 57/57 | 72/54 | place_container_plate | 96/100 | 99/94 | 98/96 |
| place_dual_shoes | 94/88 | 53/51 | 86/74 | place_empty_cup | 100/100 | 91/95 | 100/100 |
| place_fan | 96/96 | 77/74 | 90/94 | place_burger_fries | 96/99 | 95/98 | 96/98 |
| place_mouse_pad | 83/89 | 58/62 | 86/82 | place_object_basket | 89/88 | 81/72 | 76/82 |
| place_object_scale | 90/97 | 72/74 | 90/92 | place_object_stand | 90/94 | 78/85 | 92/98 |
| place_phone_stand | 97/99 | 85/88 | 94/100 | move_pillbottle_pad | 100/99 | 69/74 | 100/94 |
| place_shoe | 96/99 | 84/87 | 84/96 | put_bottles_dustbin | 95/90 | 65/65 | 92/88 |
| put_object_cabinet | 94/89 | 80/68 | 90/82 | rotate_qrcode | 93/89 | 72/84 | 82/86 |
| scan_object | 89/92 | 60/52 | 72/74 | shake_bottle | 100/100 | 100/99 | 100/100 |
| shake_bottle_horiz. | 100/100 | 100/98 | 100/100 | stack_blocks_three | 95/97 | 65/67 | 88/88 |
| stack_blocks_two | 100/100 | 94/91 | 100/100 | stack_bowls_two | 92/98 | 91/97 | 92/96 |
| stamp_seal | 90/94 | 60/63 | 72/90 | **Average** | **91.8** | **76.4** | **88.8/89.1** |

## G Additional LIBERO and LIBERO-Plus Results

**LIBERO.** Table 9 reports the suite-level performance on standard LIBERO. Each suite contains 10 tasks, with 50 evaluation rollouts per task.

**LIBERO-Plus.** Following Faster-WAM (Zhao et al., 2026), we evaluate the corresponding LIBERO checkpoints zero-shot across seven distribution shifts: camera viewpoint, robot initialization, language instruction, lighting, background, sensor noise, and object layout. Table 10 reports the detailed results.

Table 9: Suite-level success rates (%) on the four standard LIBERO suites. Baseline results follow the corresponding papers and official releases. The best and second-best results in each column are shown in bold and underlined, respectively.

| Type | Method | Spatial | Object | Goal | Long | Avg. |
|---|---|---|---|---|---|---|
| VLA | OpenVLA (Kim et al., 2025b) | 84.7 | 88.4 | 79.2 | 53.7 | 76.5 |
| | OpenVLA-OFT (Kim et al., 2025a) | 97.6 | 98.4 | 97.9 | 94.5 | 97.1 |
| | VLA-Adapter (Wang et al., 2026) | 97.8 | 99.2 | 97.2 | 95.0 | 97.3 |
| | $\pi_0$ (Black et al., 2024) | 96.8 | 98.8 | 95.8 | 85.2 | 94.1 |
| | $\pi_{0.5}$ (P. I. et al., 2025) | <u>98.8</u> | 98.2 | 98.0 | 92.4 | 96.9 |
| WAM | Fast-WAM (Yuan et al., 2026) | 98.2 | **100.0** | 97.0 | 95.2 | 97.6 |
| | LingBot-VA (Li et al., 2026b) | 98.5 | 99.6 | 97.2 | **98.5** | <u>98.5</u> |
| | Faster-WAM (Zhao et al., 2026) | **99.6** | <u>99.8</u> | <u>98.2</u> | <u>98.2</u> | **99.0** |
| | Light-WAM (Li et al., 2026c) | 98.2 | 99.6 | 97.8 | 93.0 | 97.2 |
| | **WAM-Adapter (ours)** | 98.2 | 99.6 | **98.8** | 97.4 | **<u>98.5</u>** |

Table 10: Detailed success rates (%) on LIBERO-Plus across seven distribution shifts under the Faster-WAM evaluation protocol (Zhao et al., 2026). P.T. denotes embodied pretraining.

| Method | P.T. | Camera | Robot | Lang. | Light | Backg. | Noise | Layout | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| 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., 2025a) | ✓ | 56.4 | 31.9 | 79.5 | 88.7 | 93.3 | 75.8 | 74.2 | 69.6 |
| $\pi_0$ (Black et al., 2024) | ✓ | 13.8 | 6.0 | 58.8 | 85.0 | 81.4 | 79.0 | 68.9 | 53.6 |
| $\pi_0$-Fast (Pertsch et al., 2025) | ✓ | 65.1 | 21.6 | 61.0 | 73.2 | 73.2 | 74.4 | 68.8 | 61.6 |
| WorldVLA (Cen et al., 2025) | ✓ | 0.1 | 27.9 | 41.6 | 43.7 | 17.1 | 10.9 | 38.0 | 25.0 |
| Fast-WAM (Yuan et al., 2026) | × | 18.8 | 45.7 | 70.1 | 83.2 | 45.7 | 29.8 | 62.7 | 49.1 |
| Joint-WAM (Zhao et al., 2026) | × | 37.5 | 64.5 | 93.0 | 95.0 | 55.9 | 47.3 | 79.6 | 66.3 |
| Faster-WAM (Zhao et al., 2026) | × | 53.8 | 71.6 | 94.7 | 96.3 | 61.3 | 63.6 | 79.1 | <u>73.6</u> |
| Light-WAM (Li et al., 2026c) | × | 71.75 | 23.44 | 63.32 | 76.97 | 34.27 | 68.57 | 56.28 | 55.58 |
| **WAM-Adapter (ours)** | × | **79.93** | 49.82 | 91.46 | 95.51 | 52.25 | **87.62** | 76.38 | **75.17** |

## H Real-World Experimental Details

For the real-world experiments, both WAM-Adapter and Light-WAM are trained using the same task-specific demonstration data and evaluation protocol. Each task contains 200 demonstration trajectories, with 5% of the data held out for validation. Models use Wan2.1-T2V-1.3B as the backbone and are trained for 150K steps with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$), an initial learning rate of $10^{-4}$, cosine learning-rate decay, 1,000 warmup steps, weight decay of $0.01$, gradient clipping at $1.0$, BF16 precision, and seed 42. LoRA uses rank 64, scaling factor $\alpha = 128$, and zero dropout. The input consists of three camera views, a 14-dimensional proprioceptive state, and a task instruction, with the concatenated visual input represented on a $384 \times 320$ canvas. The policy predicts a 32-step action chunk with 14-dimensional actions. Training uses eight NVIDIA A800 GPUs with a micro-batch size of 16 per GPU and gradient accumulation of one. We evaluate the 150K checkpoint for 20 real-world trials per task.

# I REAL-WORLD ROLLOUTS

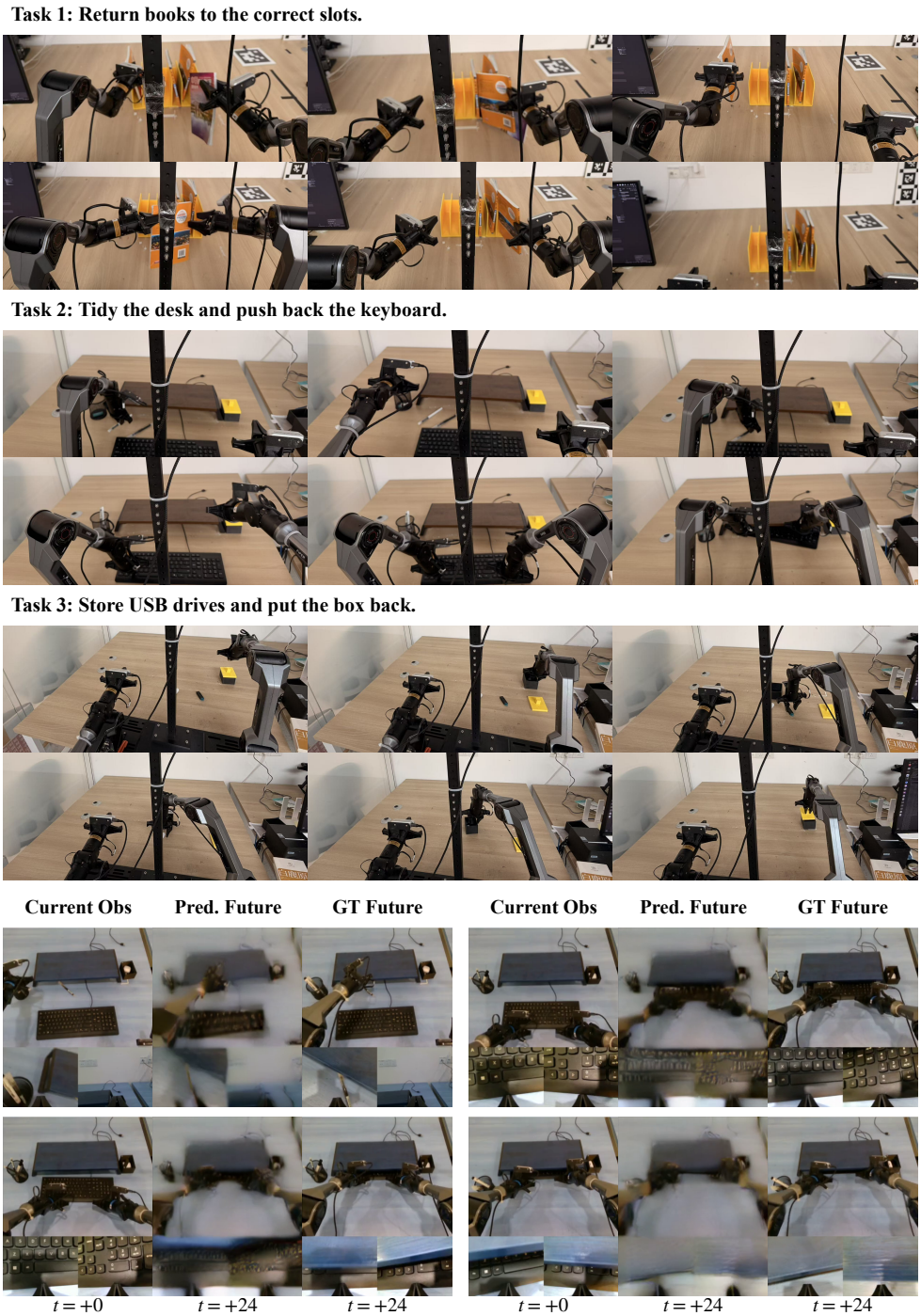


Figure 8: **Additional real-world task sequences and future predictions.** Top: third-person snapshots of the three tasks, read left to right and top to bottom within each task. Bottom: current observations, WAM-Adapter predictions, and ground-truth future observations.