FORWARD DYNAMICS MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models and world action models (WAMs) have advanced robotic policy learning by building on pretrained vision-language models (VLMs) and video generative models (VGMs), respectively. Yet VLAs learn without explicit supervision from future world states, while WAMs that condition actions on predicted futures incur extra inference cost and let visual prediction errors propagate into actions. We present **Forward Dynamics Models** (FDM), a family of robotic action models that jointly predicts actions and future world states through an *action-to-future* paradigm. Future prediction is conditioned on action representations, while action prediction remains independent of predicted futures. End-to-end training lets ground-truth future states supervise the policy through these action representations. At inference, FDM skips future generation and runs as efficiently as a standard VLA. The paradigm is backbone-agnostic, and we instantiate it with both VLM and VGM backbones. Experiments in simulation and on real robots demonstrate the effectiveness of FDM across both backbone families, achieving **99.4%** average success on LIBERO and gains of up to **13** percentage points over VLA and WAM methods on real-world tasks.

1 INTRODUCTION

Pretrained vision-language models (VLMs) and video generative models (VGMs) offer two complementary foundations for robotic action learning. Vision-language-action (VLA) models adapt VLMs to map current observations and task instructions directly to actions (Kim et al., 2024; Intelligence et al., 2025; Yu et al., 2026b; NVIDIA et al., 2025), following the action-imitation paradigm (Chi et al., 2025; Zhao et al., 2023) (Figure 1(a)). While effective, this paradigm learns *what* to do without explicit supervision on *what happens next*, leaving the rich signal in future world states unused. World action models (WAMs) instead build on video backbones to model visual dynamics alongside actions (Hu et al., 2024; Feng et al., 2025; Pai et al., 2025; Bi et al., 2026a). A prevalent design first predicts future world states and then infers actions from them in an inverse-dynamics manner (Li et al., 2026b; Zhao et al., 2025; Ye et al., 2026c; Ma et al., 2026b; AgiBot Research Team et al., 2026; Tian et al., 2024) (Figure 1(b)). However, future prediction is inherently error-prone, since every visual detail matters across both space and time. Placing it *before* action prediction means its errors accumulate and amplify into the actions, and future generation also adds substantial inference overhead. Other works predict future frames and actions jointly or in parallel (Ye et al., 2026c; Wang et al., 2026b), align policy features with future embeddings (Zheng et al., 2025), or train an action model and an action-conditioned world model side by side (Cen et al., 2025b). These designs motivate studying the specific contribution of future supervision through action representations, beyond supervision through shared context.

We investigate whether a policy can benefit from future supervision without ever depending on predicted futures. To this end, we introduce **Forward Dynamics Models** (FDM), a family of robotic action models built on an *action-to-future* learning paradigm. Borrowing the classical notion of forward dynamics (Jordan & Rumelhart, 1992; Agrawal et al., 2016), the mapping from state and action to the next state, FDM connects action learning with future world state prediction: given the current world state and task instruction, FDM jointly predicts actions and future world states with asymmetric conditioning. Future prediction reads action representations, while action prediction cannot access future information. The model is trained end to end, so the future-prediction loss backpropagates through these action representations into the policy. Action-to-future specifies this one-way information dependency within joint prediction. This reverses the conditioning direction of

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

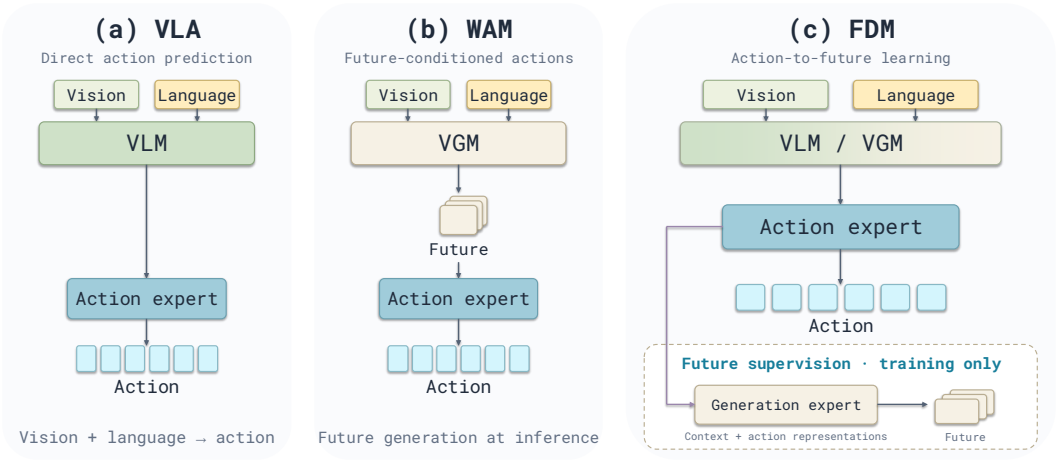


Figure 1: **Architectural comparison. (a)** VLA predicts actions directly from the current observation and instruction. **(b)** A future-conditioned WAM predicts future video and conditions action prediction on it. **(c)** FDM jointly predicts actions and future world states: future prediction reads action representations, while action prediction cannot access future information.

future-conditioned WAMs and brings several advantages: (1) *Robustness*: action prediction does not depend on predicted future world states, so errors in those predictions cannot propagate into actions; (2) *Action-grounded supervision*: ground-truth future states supervise the action representations used to generate actions, complementing action imitation with supervision from observed consequences; (3) *Efficiency*: action-only inference skips future generation while retaining the context processing required for action prediction; (4) *Flexibility*: either a VLM or a VGM can serve as the backbone. Table 1 contrasts the formulations of the three paradigms. Our analysis further confirms that, unlike future-conditioned WAMs, whose action error grows with the quality degradation of predicted futures, FDM's action error is insensitive to future-prediction quality (Section 4.4).

Our *contributions* are threefold: **(i)** We introduce FDM, a novel family of robotic action models that learns in an action-to-future manner during training, with future supervision flowing into the policy through its action representations, and requires no future prediction at inference; **(ii)** We build VLM- and VGM-based instantiations of FDM, demonstrating its backbone flexibility; **(iii)** Experiments in simulation and on real robots show that FDM achieves strong or competitive performance across VLM- and VGM-based settings, and analysis shows that it avoids the error propagation of future-conditioned designs.

2 Related Work

Vision-language-action models. VLA models adapt pretrained vision-language models to predict robot actions (Wang et al., 2026a; Yuan et al., 2026a; Ye et al., 2026b; Black et al., 2024; Yang et al., 2026; Liu et al., 2026; Yu et al., 2026a; Zhang et al., 2026b; Zheng et al., 2026; Qu et al., 2025;

Table 1: **Formulation comparison.** $C = (O_t, L)$: input context comprising the current observation O_t and task instruction L; A: action chunk; V: future world state target. WAM here denotes the future-conditioned formulation. For FDM, the formulation expresses action-to-future dependencies realized through action representations; inference requires only action prediction.

Model family	Training	Inference
VLA	$p_\theta(A \mid C)$	$p_\theta(A \mid C)$
WAM	$p_\theta(V \mid C)\, p_\theta(A \mid C, V)$	$p_\theta(V \mid C)\, p_\theta(A \mid C, V)$
FDM	$p_\theta(A \mid C)\, p_\theta(V \mid C, A)$	$p_\theta(A \mid C)$

Contributors, 2025; Cai et al., 2026b; Shukor et al., 2025; Wang et al., 2025). Recent approaches also predict future observations or their latent representations (Cen et al., 2025b; Ma et al., 2026b; Cheang et al., 2024; Cen et al., 2025a; Lv et al., 2025; Zhang et al., 2026a; Hu et al., 2026; Kang et al., 2026; Yang et al., 2025; Zhang et al., 2025), using future prediction as a learning objective or an input to action generation, as in visual chain-of-thought policies (Zhao et al., 2025). Our VLM-based FDM builds on $\pi_{0.5}$ and conditions future prediction on action representations and the current context, establishing an action-to-future dependency.

World action models. Video models provide visual dynamics representations for robot control (HaCohen et al., 2024; Ali et al., 2025; Wan et al., 2025). WAMs use these representations to condition action decoding (Hu et al., 2024; Pai et al., 2025; Ma et al., 2026c) or jointly predict video and actions (Bi et al., 2026a; Ye et al., 2026c; Li et al., 2026a; Wang et al., 2026c; Zhou et al., 2026; Tian et al., 2026; Zhu et al., 2025). Extensions combine predicted outcomes with value estimates for planning (Bi et al., 2026b; Kim et al., 2026; Chen et al., 2026), while asynchronous designs improve efficiency through different denoising budgets or prediction horizons (Guo et al., 2026; Cai et al., 2026a). Our VGM-based FDM places future prediction downstream of action representations, keeping the action path independent of predicted video. Deployment therefore omits future-video computation while retaining current-observation processing.

Joint predictive models use visual supervision for action learning (Wang et al., 2026b; Ma et al., 2026a; Ye et al., 2026a; Xu et al., 2026; Wu et al., 2023; Zheng et al., 2025; Won et al., 2025). UVA and Fast-WAM also enable action-only deployment via separate video–action decoders and structured attention, respectively (Li et al., 2025; Yuan et al., 2026b). FDM conditions future prediction on action representations and backpropagates prediction errors into them. Our context-only and stop-gradient controls isolate action conditioning and backward supervision, respectively.

3 METHOD

FDM is a family of robotic action models that jointly predicts actions and future world states. Its action-to-future paradigm uses asymmetric conditioning: future prediction reads action representations, while action prediction cannot access future information. Future prediction errors supervise action learning through these representations during training. Let $C = (O_t, L)$ denote the current observation O_t and task instruction L, A an action chunk, and V the future world state target. The current world state is observed through images; future world state targets are represented by VAE latents of future images or video in our implementations. The intended information dependencies are

$$p\left(A, V \mid C\right) = p\left(A \mid C\right) p\left(V \mid C, A\right). \tag{1}$$

Equation (1) is a conceptual statement of these dependencies; the learning mechanism is the representation-level connection defined below. In our implementations, the future-prediction branch is conditioned through action-expert hidden states, rather than through the final numerical action output. This conditioning direction holds within joint prediction and permits action-only inference that skips future generation. Future prediction remains optional for visual rollout. We first specify the representation and supervision path, then describe how our VLM- and VGM-based instantiations realize it.

3.1 ACTION-TO-FUTURE REPRESENTATIONS AND SUPERVISION

Let H_a denote the action representations read by the future-prediction branch, and let θ_a denote the parameters that produce them. In our implementations, these are action-expert hidden states and parameters, respectively. Both instantiations use action flow matching, with $X_a = (A_{\tau_a}, \tau_a)$ comprising the noised action chunk A_{τ_a} and its action flow-matching time $\tau_a \in [0, 1]$. The action velocity prediction and future objective take the form

$$H_a = f_{\theta_a}(C, X_a), \qquad \widehat{u}_a = D_\psi(H_a), \qquad \mathcal{L}_{\text{world}} = \ell(G_\phi(C, H_a, \ldots), V), \tag{2}$$

where D_ψ is the action velocity head and ℓ denotes the appropriate future-prediction objective, with noise and time dependencies suppressed in this schematic expression. The generation expert reads H_a; the velocity output $\widehat{u}_a$ is used to update the action chunk during denoising. For the VLM-based FDM, G_ϕ predicts future-image latents using the MSE objective in Equation (4); for the VGM-based

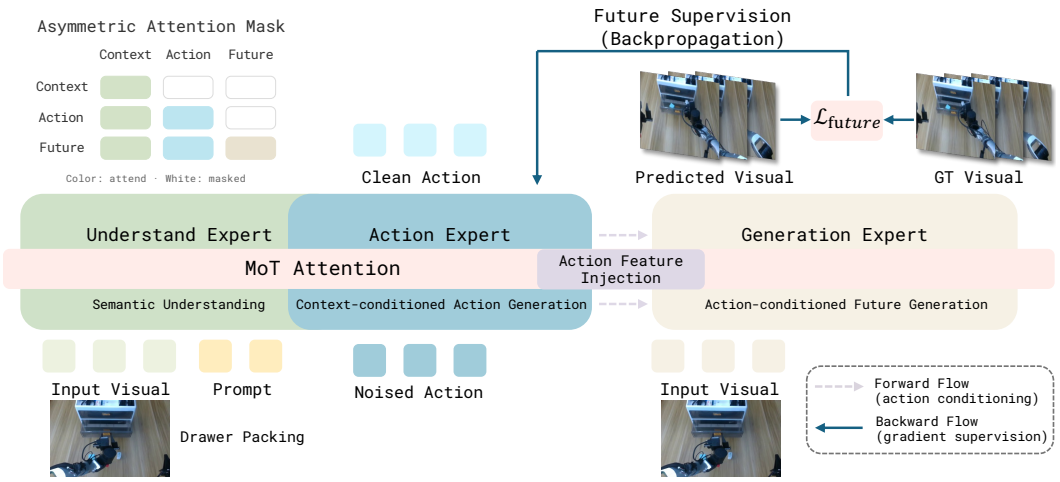


Figure 2: **Overview of Forward Dynamics Model.** The understanding expert extracts task-relevant context from visual observations and the instruction to guide the action expert. The generation expert predicts future observations conditioned on this context and action-expert hidden states. Future prediction errors backpropagate through the action representations, complementing action imitation with supervision from observed action consequences. The asymmetric attention mask allows future tokens to attend to context and action tokens while preventing context and action tokens from attending to future tokens. At deployment, future prediction is skipped while the context processing needed for action generation is retained.

FDM, it is the velocity predictor v_ϕ trained using the flow-matching objective in Equation (6). The future objective $\mathcal{L}_{\text{world}}$ contributes the gradient

$$\nabla_{\theta_a} \mathcal{L}_{\text{world}}\big|_{H_a} = \left(\frac{\partial H_a}{\partial \theta_a}\right)^{\top} \nabla_{H_a} \mathcal{L}_{\text{world}}. \tag{3}$$

This path supervises the internal features used for action generation. To isolate its contribution, Section 4.3 replaces only the generation-facing H_a with $\text{sg}(H_a)$, which preserves its values but blocks gradients, and measures the change in closed-loop policy success. Future-latent perturbations examine error propagation to actions at inference, while gradient checks examine the backward supervision signal.

Our two instantiations implement these dependencies with a mixture-of-transformers (MoT) architecture (Liang et al., 2024) (Figure 2). Each expert processes its tokens with self-attention, while MoT connects the experts through their hidden states. The generation expert reads action-expert hidden states as conditions for future prediction; the action expert cannot read future visual states. This restriction also applies to indirect paths: observation or understanding features accessible to the action expert must remain independent of generation-branch tokens. Thus, future supervision can train the policy without making future prediction an input to action generation.

3.2 VLM-BASED INSTANTIATION

The VLM-based FDM, FDM-VLM, uses an understanding expert, an action expert, and a generation expert. The understanding expert plays the role of the VLM in a conventional VLA, extracting semantic features from the current images and instruction. The action expert reads these understanding features to generate actions. The generation expert combines the Wan VAE latents (Wan et al., 2025) of historical and current images with understanding features and action-expert hidden states to directly predict the clean latent of one future image per camera view.

Let V denote the future-image latents across camera views and $\widehat{V} = G_\phi(C, H_a, \ldots)$ their prediction. We minimize the mean squared error between predicted and target latents:

$$\mathcal{L}_{\text{img}} = \mathbb{E}\left[\text{MSE}(\widehat{V}, V)\right], \tag{4}$$

where MSE averages the per-view mean squared errors across camera views.

3.3 VGM-BASED INSTANTIATION

The VGM-based FDM, FDM-VGM, uses an action expert and a generation expert. The video backbone serves both the understanding and generation roles: it extracts current-observation features for action generation and predicts action-conditioned future video. The action expert reads the generation expert's current-observation features, while future-video tokens read action-expert hidden states through MoT. Although observation and future tokens belong to the same generation expert, their attention permissions differ: the features available to the action policy cannot incorporate future information.

Here, V denotes the clean future-video latents encoded by the Wan VAE, and C provides the current observation and task instruction. During training, the current-frame latent remains fixed while the future latents are noised for flow matching. Using the convention

$$V_{\tau_v} = (1 - \tau_v)V + \tau_v \epsilon_v, \qquad \epsilon_v \sim \mathcal{N}(0, I), \tag{5}$$

the generation expert predicts a velocity field and minimizes

$$\mathcal{L}_{\text{vid}} = \mathbb{E}\left[\left\| v_\phi(V_{\tau_v}, \tau_v; C, H_a) - (\epsilon_v - V) \right\|_2^2\right], \tag{6}$$

where $\tau_v \in [0, 1]$ is the video flow-matching time, independent of the action flow-matching time τ_a, and the loss is evaluated only on future latents. Future-video tokens interact through video attention to model temporal content.

3.4 JOINT TRAINING AND ACTION-ONLY INFERENCE

Both instantiations combine the action flow-matching objective $\mathcal{L}_{\text{act}}$ with future supervision:

$$\mathcal{L}_{\text{total}} = \lambda_a \mathcal{L}_{\text{act}} + \lambda_v \mathcal{L}_{\text{world}}, \tag{7}$$

where λ_a and λ_v weight action learning and future supervision, respectively, and $\mathcal{L}_{\text{world}}$ is $\mathcal{L}_{\text{img}}$ for the FDM-VLM and $\mathcal{L}_{\text{vid}}$ for the FDM-VGM. Real-world training settings are provided in Section C. The action objective trains the policy to reproduce demonstrated actions, while the future objective supervises the action-expert features through their MoT connection to the generation expert.

Action-only inference. At inference, both policies generate action chunks from noise conditioned on the current observation and task instruction, without predicting future visual states. This action-only path avoids compounding errors from future visual predictions and removes the computational overhead of future generation, supporting high-frequency reactions to new observations.

4 EXPERIMENTS

We evaluate FDM across two backbone families in simulation and the real world. Beyond task success, we analyze how future supervision contributes to action learning and how the action-to-future design affects inference behavior and computational cost.

4.1 EXPERIMENTAL SETUP

All experiments are conducted on 8 NVIDIA H100 GPUs. Appendix C reports the VLM- and VGM-based FDM training hyperparameters used in the real-world experiments.

Two backbone families. We instantiate FDM with two backbone families to assess future supervision across different policy architectures. For the VLM-based FDM, we extend $\pi_{0.5}$ (Intelligence et al., 2025) with a randomly initialized generation expert that has the same network architecture as its action expert. For the VGM-based FDM, we add a randomly initialized action expert to Wan2.2-5B (Wan et al., 2025). In both instantiations, MoT (Liang et al., 2024) enables the future-prediction branch to read action-expert hidden states, following the directional information flow in Section 3.

Table 2: **Results on LIBERO.** Success rates (%). P.T.: robot policy pretraining. Ranks within each backbone family (Tables 2–3): best / second best.

Method	P.T.	Spatial	Object	Goal	Long	Avg.
VLM-based						
π_0 (Black et al., 2024)	✓	96.8	98.8	95.8	85.2	94.1
$\pi_{0.5}$ (Intelligence et al., 2025)	✓	98.8	98.2	98.0	92.4	96.9
FDM-VLM (Ours)	✗	99.2	100.0	99.8	98.6	99.4
VGM-based						
Motus (Bi et al., 2026a)	✓	96.8	99.8	96.6	97.6	97.7
LingBot-VA (Li et al., 2026a)	✓	98.5	99.6	97.2	98.5	98.5
FDM-VGM (Ours)	✗	98.8	100.0	99.2	94.2	98.1

Table 3: **Results on RoboTwin 2.0.** Success rates (%) over 50 tasks. Avg. averages Clean and Rand.

Method	P.T.	Clean	Rand.	Avg.
VLM-based				
π_0 (Black et al., 2024)	✓	65.92	58.40	62.16
$\pi_{0.5}$ (Intelligence et al., 2025)	✓	82.74	76.76	79.75
FDM-VLM (Ours)	✗	81.54	80.86	81.20
VGM-based				
Motus (Bi et al., 2026a)	✓	88.66	87.02	87.8
LingBot-VA (Li et al., 2026a)	✓	92.90	91.50	92.20
FDM-VGM (Ours)	✗	88.84	89.62	89.23

Benchmarks and real-world tasks. We evaluate task success in simulation on LIBERO (Liu et al., 2023) and RoboTwin 2.0 (Chen et al., 2025). LIBERO covers the Spatial, Object, Goal, and Long suites, while RoboTwin evaluates bimanual manipulation under clean and randomized conditions. On both simulation benchmarks, we compare FDM-VLM with the VLA methods π_0 and $\pi_{0.5}$, and FDM-VGM with the WAM methods Motus (Bi et al., 2026a) and LingBot-VA (Li et al., 2026a). Both FDM instantiations are trained without robot policy pretraining (P.T.). To assess performance beyond simulation, we use a dual-arm ARX-X5 robot for five tasks: loading a drawer with a charger and a block, then closing it (Drawer); packing a scarf and a flower into a box and closing it (Box); inserting two shuttlecocks into a tube (Shuttlecock); stacking two cups on a three-cup base (Cup); and stacking two small blocks on a larger block (Block). The methods compared in the real world are π_0 and $\pi_{0.5}$ for the VLM-based model, and DreamZero and a future-conditioned WAM (denoted WAM) for the VGM-based model. This WAM shares the architecture of FDM-VGM, differing only in its attention mask, which reverses the direction of cross-expert attention. We use the same WAM implementation for real-world task success (Section 4.2), future-perturbation analysis (Section 4.4), and efficiency evaluation (Section 4.5). The hardware setup, teleoperation interface, and detailed task descriptions are provided in Appendix D.

4.2 Q1: Does future supervision improve task success?

We assess the effectiveness of future supervision by comparing each FDM instantiation with VLA and WAM methods, first on LIBERO and RoboTwin and then on five real-robot tasks. This evaluation examines whether the benefit holds across policy architectures and manipulation settings.

LIBERO. The VLM-based FDM improves over $\pi_{0.5}$ across all four suites, raising average success from 96.9% to 99.4% and LIBERO-Long success from 92.4% to 98.6% (Table 2). The VGM-based FDM achieves 98.1% average success, compared with 97.7% for Motus and 98.5% for LingBot-VA.

RoboTwin. The VLM- and VGM-based FDM achieve average success rates of 81.20% and 89.23%, respectively, improving over $\pi_{0.5}$ (79.75%) and Motus (87.8%) (Table 3). Both improve under randomized conditions. Appendix E reports results for all 50 tasks.

Real-world manipulation. We next evaluate whether the gains observed in simulation carry over to physical manipulation. The VLM-based FDM achieves 96% average success, improving over $\pi_{0.5}$ (83%) on all five tasks, while the VGM-based FDM reaches 83%, compared with 78% for the architecture-matched WAM and 71% for DreamZero (Figure 3). Both instantiations achieve 100% success on Cup and Block, demonstrating strong performance on the two stacking tasks within an evaluation that also covers sequential object handling, cloth manipulation, and insertion.

4.3 Q2: Does backpropagation through action representations improve control?

Backpropagating future prediction errors through H_a improves closed-loop task success in both backbone families. To isolate this contribution, we ablate the action-to-future connection on five real-robot tasks. We compare four variants that separate the presence of a future objective, forward access to action representations, and backward supervision through that interface (Table 4). **A. Action-only**

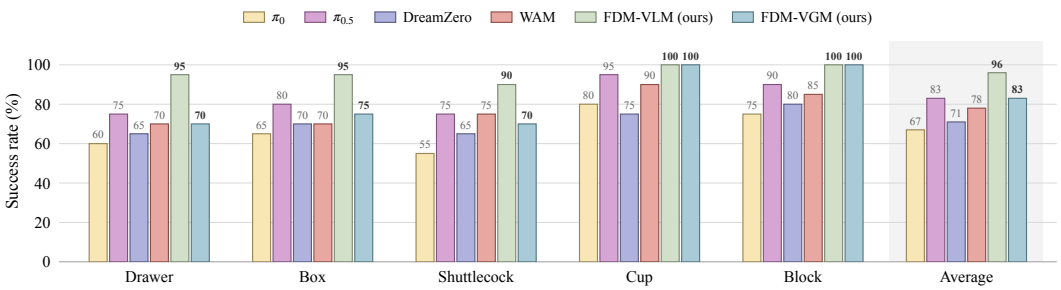


Figure 3: **Main results on the five real-robot tasks.** Per-task and average success rates, 20 trials per method per task. Both FDM variants outperform the corresponding comparison methods on average.

trains with action imitation alone. **B. Context-only future** adds the future objective but prevents generation-branch tokens from reading action representations. **C. Stop-gradient action conditioning** lets the future-prediction branch read $\mathrm{sg}(H_a)$, which preserves values but blocks gradients. **D. Full FDM** lets the future-prediction branch read H_a and backpropagate through it.

Controlled protocol. Within each backbone, we compare all four variants under matched training and evaluation settings, using the same training seeds. We measure closed-loop success on the five real-robot tasks with action-only inference. Appendix C.3 details the controlled settings and stop-gradient implementation.

Backward future supervision improves control (D vs. C). Full FDM raises average success from 89% to 96% for the VLM-based model and from 78% to 83% for the VGM-based model, gains of 7 and 5 percentage points over Stop-gradient, respectively. In each backbone family, Full improves success on four tasks and matches Stop-gradient on the remaining task. For the VLM-based model, the largest gains occur on Drawer and Box, at 15 and 10 percentage points; for the VGM-based model, the largest gain is on Cup, at 10 points. C and D retain the same forward action conditioning: for identical parameters, inputs, and noise, $\mathrm{sg}(H_a)$ and H_a yield identical forward predictions. Their difference is whether future-loss gradients pass through the action-representation interface during training. The resulting success-rate gains therefore support the value of this backward supervision for the deployed action-only policy.

The action-to-future connection adds to visual co-training (A, B, and D). Adding a context-only future objective increases average success from 83% to 87% for the VLM-based model and from 66% to 72% for the VGM-based model (B vs. A). Full FDM further improves over this control by 9 and 11 percentage points, respectively (D vs. B), with higher success on all five tasks in both backbone families. Relative to action-only training, the total gains reach 13 points for the VLM-based model and 17 points for the VGM-based model (D vs. A). Thus, visual co-training provides a useful learning signal, and connecting future prediction to action representations yields additional gains. D vs. B captures the combined contribution of forward conditioning and backward supervision, while D vs. C isolates the benefit of backpropagation through H_a.

4.4 Q3: Does future-prediction error propagate to actions?

We examine whether perturbing the predicted future of the same WAM evaluated for real-world task success in Section 4.2 changes its action error, and contrast this dependence with FDM's action-to-future design. Appendix A.5 details the WAM implementation used here and in Section 4.5. We conduct open-loop evaluation across simulation and real-world tasks—RoboTwin, LIBERO, Drawer, Box, Shuttlecock, Cup, and Block—using the full validation set for each. For each validation sample, we hold the observation, task instruction, and sampling randomness fixed across experimental conditions.

Perturbing the WAM's future prediction. At an intermediate stage of joint denoising, we inject Gaussian noise once into the future-video latent, leaving the current observation and noised action chunk unchanged, and then resume normal sampling. We evaluate perturbation strengths $\alpha \in$

Table 4: **Action-to-future ablation on five real-robot tasks.** Future: future supervision; Reads H_a: action-representation conditioning; Backprop: future-loss gradients through H_a.

Model	Variant	Action-to-future pathway			Success rate (%) ↑					
		Future	Reads H_a	Backprop	Drawer	Box	Shuttlecock	Cup	Block	Average
VLM	A. Action-only	✗	✗	✗	75	80	75	95	90	83
	B. Context-only future	✓	✗	✗	85	80	80	95	95	87
	C. Stop-gradient	✓	✓	✗	80	85	85	95	100	89
	D. FDM (Ours)	✓	✓	✓	95	95	90	100	100	96
VGM	A. Action-only	✗	✗	✗	50	60	55	85	80	66
	B. Context-only future	✓	✗	✗	60	65	55	90	90	72
	C. Stop-gradient	✓	✓	✗	70	70	65	90	95	78
	D. FDM (Ours)	✓	✓	✓	70	75	70	100	100	83

$\{0, 0.1, 0.25, 0.5, 1.0\}$, where $\alpha = 0$ is the unperturbed condition, and report the percentage increase in action error relative to this condition.

Future perturbations affect WAM actions. Figure 4 shows that the WAM's action error generally increases with perturbation strength across simulation and real-world tasks. At $\alpha = 1.0$, action error rises by 4.44% on LIBERO and 5.52% on RoboTwin; all five real-world tasks also show increases, ranging from 0.22% on Box to 1.17% on Cup. Small perturbations yield slight decreases on LIBERO and Box, but all seven settings exhibit higher action error at $\alpha \geq 0.5$, with the largest increase at $\alpha = 1.0$. These results show that perturbations to the generated future can affect the WAM's action error even when the observation and noised action chunk are unchanged at the intervention step.

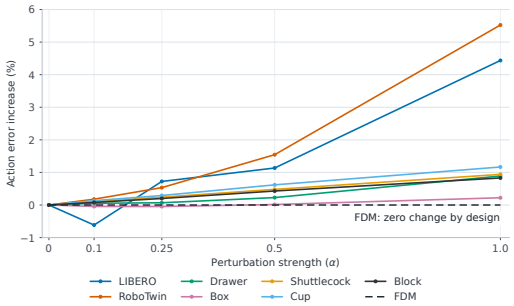


Figure 4: **Action sensitivity to future perturbations.** Solid curves show the percentage increase in action error relative to $\alpha = 0$.

FDM's action inference is independent of future quality. In FDM, the future-prediction branch reads action representations, while the action path never reads generated future states. Consequently, for fixed model parameters, inputs, and action-sampling randomness, perturbing the generated future leaves action predictions unchanged, regardless of future-generation quality. The dashed zero line in Figure 4 denotes this architectural invariance: zero change in action error. Thus, FDM benefits from future supervision during training (Section 4.3) while keeping action inference independent of future-generation errors.

Q4: Are future and action gradients aligned?

We examine whether future supervision provides a locally compatible training signal for action imitation. We compare action-imitation gradients with the future-loss gradient contribution propagated through the action-to-future interface, measuring their cosine similarity on shared-context key/value (K/V) projections. This analysis isolates the backward contribution through action representations, rather than using the total future-loss gradient.

Figure 5 reports gradient cosine similarity over training progress for FDM-VLM and FDM-VGM. FDM-VLM exhibits predominantly positive alignment across the evaluated settings, with only limited early-stage deviations. Most FDM-VGM settings also exhibit positive alignment after initial fluctuations. These results indicate broadly compatible gradient directions between future supervision and action imitation in this analysis. They support the use of the action-to-future connection to provide an auxiliary training signal for action learning, complementing the policy-level evidence in Section 4.3.

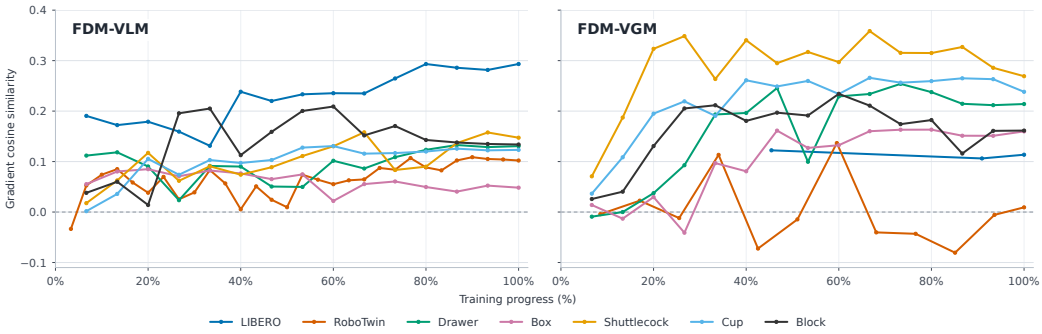


Figure 5: **Alignment between future and action gradients.** Cosine similarity between action-imitation gradients and future-loss gradients propagated through the action-to-future interface.

4.5 Q5: What are the training cost and inference efficiency?

We compare training step time and end-to-end action-chunk latency across FDM and the VLA and WAM paradigms within each backbone family under matched settings (Table 5). We use $\pi_{0.5}$ as the VLA method, while the evaluated future-conditioned WAM jointly supervises actions and futures during training and generates futures for action prediction at inference. We evaluate the same architecture-matched WAM used for real-world task success in Section 4.2 and future perturbations in Section 4.4. Future supervision increases training step time for the VLM-based policy while maintaining inference latency comparable to $\pi_{0.5}$. For the VGM-based policy, training cost remains essentially unchanged, while action-only inference substantially reduces latency by removing future generation from the action prediction path. Specifically, VLM-based training step time increases from 0.444 to 0.631 s ($+42.1\%$), while inference latency remains comparable at 294.1 versus 298.0 ms/chunk. VGM-based training step time is nearly unchanged at 0.973 versus 0.975 s, while inference latency drops from 715.4 to 420.3 ms/chunk, yielding a $1.7\times$ speedup.

Table 5: **Training cost and inference efficiency.** Comparisons use matched settings within each backbone family. (a) $\pi_{0.5}$ (VLA) uses action supervision; WAM uses joint action–future supervision. (b) Only WAM requires future generation for action prediction. WAM denotes the same future-conditioned method evaluated in Figure 3. Relative metrics use $\pi_{0.5}$ for FDM-VLM and WAM for FDM-VGM.

(a) Training cost

Method	Step time ↓ (s/step)	Throughput ↑ (samples/s)	Memory ↓ (GiB/GPU, peak)	Overhead ↓ (%)
$\pi_{0.5}$	0.4436	288.58	67.08	—
FDM-VLM	0.6305	203.01	72.04	+42.14
WAM	0.9754	131.23	66.80	—
FDM-VGM	0.9729	131.56	66.80	−0.26

(b) Inference efficiency

Method	Latency ↓ (ms/chunk)	Speedup ↑ (×)	Reduction ↑ (%)
$\pi_{0.5}$	298.00	1.00	0.00
FDM-VLM	294.13	1.01	1.30
WAM	715.36	1.00	0.00
FDM-VGM	420.34	1.70	41.24

5 Conclusion

We introduced FDM, a family of robotic action models built around action-to-future learning. Actions and future world states are jointly predicted with asymmetric conditioning: future prediction reads action representations, while action prediction remains independent of predicted futures. Future prediction errors supervise these action representations through backpropagation. Ground-truth future states thus supervise the policy during training, while action-only inference skips future generation. Our VLM- and VGM-based instantiations realize this principle via future-image latent regression and future-video flow matching, respectively. They achieve strong LIBERO performance and improve success on RoboTwin and five real-world tasks. Controlled ablations and gradient analysis examine backward future supervision, while future-latent perturbations test action sensitivity to prediction errors at inference. The VLM-based FDM retains comparable action latency, while the VGM-based FDM reduces latency relative to the evaluated future-conditioned WAM.

AI USE STATEMENT

Generative AI tools were used only to assist with writing, LaTeX formatting, and information retrieval during manuscript preparation. They were not used to develop the method, design experiments, implement research code, process data, or interpret results. The authors manually reviewed the AI-assisted content and checked the references. The authors take responsibility for the final content of this work, including any text, claims, or artifacts prepared with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 describes the action–future information flow, the VLM- and VGM-based instantiations, their training objectives, and action-only inference. Section 4.1 identifies the backbone families, comparison methods, and evaluation benchmarks, while Section 4.3 describes the supervision variants and matched comparison design. Appendix D documents the real-robot hardware, camera configuration, teleoperation interface, and task descriptions. Appendix C reports the VLM- and VGM-based FDM training hyperparameters used in the real-world experiments, including optimization settings, batch sizes, input resolutions, and action prediction horizons.

REFERENCES

AgiBot Research Team, Renhang Liu, Wenzhi Zhao, Zhuo Yang, Liliang Chen, Pengfei Zhou, Shengcong Chen, Guanghui Ren, Youlun Peng, Rongjun Jin, Nan Wang, Sukai Wang, Xindong He, Jinyuan Feng, Ziyu Xiong, Linqing Zhong, Yifei Wei, Feng Han, Long Zhang, Da Huang, Nanshu Zhao, Chenghao Yin, Mo Wu, Zhaodong Yan, Kongtao Hu, Yuxiang Yan, Aogelijiang Niyazi, Yu Fang, Jia Zeng, Lizhu Meng, Daizhen Lv, Haoyu Cao, Zhiwen Hou, Lianjin Ye, Yuehan Niu, Zhikai Cai, Xuan Hu, Hui Min, Xiongfeng Cai, Yue Liao, Jing Wu, Soujanya Poria, Ye Li, Sanping Zhou, and Maoqing Yao. Ge-act 2.0: Pretraining and scaling a world-action model for robotic manipulation, 2026.

Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint*, 2025.

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 *CVPR*, 2026a.

Hongzhe Bi, Zihao Zhou, Yihang Tang, Jingrui Pang, Shuhe Huang, Haitian Liu, Runqing Wang, Shuai Huang, Yichen Wang, Yiming Cheng, et al. Motus2: A self-evolving general world model for dexterous manipulation. *arXiv preprint*, 2026b.

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*, 2024.

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

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint*, 2026b.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Bohan Hou, Yuming Jiang, Jiayan Guo, Xin Li, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint*, 2025a.

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*, 2025b.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint*, 2024.

Jintao Chen, Peidong Jia, Qingpo Wuwu, Jiaming Liu, Mengfei Du, Chun-Kai Fan, Xiaowei Chi, Hao Chen, Chengyu Bai, Zezhong Qian, et al. Mv-wam: Manifold-aware world action model with value augmentation. *arXiv preprint*, 2026.

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*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2025.

InternVLA-M1 Contributors. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint*, 2025.

Yao Feng, Hengkai Tan, Xinyi Mao, Chendong Xiang, Guodong Liu, Shuhe Huang, Hang Su, and Jun Zhu. Vidar: Embodied video diffusion model for generalist manipulation. *arXiv preprint*, 2025.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint*, 2026.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint*, 2024.

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*, 2024.

Yucheng Hu, Jianke Zhang, Yuanfei Luo, Yanjiang Guo, Xiaoyu Chen, Xinshu Sun, Kun Feng, Qingzhou Lu, Sheng Chen, Yangang Zhang, et al. Bagelvla: Enhancing long-horizon manipulation via interleaved vision-language-action generation. *arXiv preprint*, 2026.

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

Michael I. Jordan and David E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, 1992.

Miracle Kang, Lights Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Sylas Chen, Shawn Qin, Yinan Zheng, Jinliang Zheng, et al. X-tokenizer: A multimodal action tokenizer for vision-language-action pretraining. *arXiv preprint*, 2026.

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

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. *arXiv preprint*, 2026.

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*, 2026a.

Shalfun Li, Victor Yao, Charles Yang, Truth Qu, Regis Cheng, Ryan Yu, Howard Lu, Newton Von, Vincent Chen, Yohann Tang, et al. Wall-wm: Carving world action modeling at the event joints. *arXiv preprint*, 2026b.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model, 2025. *arXiv preprint*, 2025.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint*, 2024.

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

Zhe Liu, Jinghua Hou, Yuxiang Lu, Zhenya Yang, Xianzhe Fan, Junwei Luo, Junyi Li, Ruihua Han, Zhi Hou, and Hengshuang Zhao. Streampi: Streaming multimodal temporal modeling for vision-language-action models. *arXiv preprint*, 2026.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint*, 2025.

Fulong Ma, Daojie Peng, Wenjun Yue, Jiahang Cao, Bintao Wang, Qiang Zhang, and Jun Ma. Geosem-wam: Geometry- and semantic-aware world action models. *arXiv preprint*, 2026a.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1.5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint*, 2026b.

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*, 2026c.

NVIDIA, 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, 2025.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *RSS*, 2025.

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

Shuai Tian, Yupeng Zheng, Yuhang Zheng, Songen Gu, Yujie Zang, Yuxing Qin, Weize Li, Haoran Li, Wenchao Ding, and Dongbin Zhao. Vt-wam: Visual-tactile world action model for contact-rich manipulation. *arXiv preprint*, 2026.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. *arXiv preprint*, 2024.

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*, 2025.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, et al. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint*, 2026a.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint*, 2025.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *ICLR*, 2026b.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, et al. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint*, 2026c.

John Won, Kyungmin Lee, Huiwon Jang, Dongyoung Kim, and Jinwoo Shin. Dual-stream diffusion for world-model augmented vision-language-action model. *arXiv preprint*, 2025.

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. *arXiv preprint*, 2023.

Gangwei Xu, Qihang Zhang, Jiaming Zhou, Xing Zhu, Yujun Shen, Xin Yang, and Yinghao Xu. Next forcing: Causal world modeling with multi-chunk prediction. *arXiv preprint*, 2026.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint*, 2026.

Yi Yang, Xueqi Li, Yiyang Chen, Jin Song, Yihan Wang, Zipeng Xiao, Jiadi Su, You Qiaoben, Pengfei Liu, and Zhijie Deng. Mantis: A versatile vision-language-action model with disentangled visual foresight. *arXiv preprint*, 2025.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigaworld-policy: An efficient action-centered world–action model. *arXiv preprint*, 2026a.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla-α: Reducing complexity in vision-language-action systems. *arXiv preprint*, 2026b.

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, 2026c.

Qize Yu, Jiadi You, Yuran Wang, Jiaqi Liang, Bowen Ping, Yang Tian, Yue Chen, Minghong Cai, Zeying Gong, Ruihai Wu, et al. Affordancevla: A vision-language-action model empowering action generation through affordance-aware understanding. *arXiv preprint*, 2026a.

Ryan Yu, Pushi Zhang, Starrick Liu, Brae Liu, Miracle Kang, Shalfun Li, Lights Shi, Ellie Ma, Ping Yang, Chris Pan, et al. Wall-oss-0.5 technical report. *arXiv preprint*, 2026b.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint*, 2026a.

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

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint*, 2025.

Jianke Zhang, Yuanfei Luo, Yucheng Hu, Xiaoyu Chen, Yanjiang Guo, Ziyang Liu, Hongbin Xu, Tian Lan, and Jianyu Chen. Uam: A dual-stream perspective on forgetting in vla training. *arXiv preprint*, 2026a.

Lingfeng Zhang, Xiaoshuai Hao, Yingbo Tang, Lei Zhou, Shuyi Zhang, Jinkun Liu, Hongsheng Li, Chenhao Zhang, Qiang Zhang, Hangjun Ye, et al. Onevla: A unified framework for embodied tasks. *arXiv preprint*, 2026b.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *CVPR*, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint*, 2023.

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 *ICLR*, 2026.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint*, 2025.

Jiaming Zhou, Qihang Zhang, Gangwei Xu, Cunxin Fan, Yujie Zhao, Ruilin Wang, Yiming Luo, Shuai Yang, Xing Zhu, Yujun Shen, et al. Zero-wam: In-context world-action modeling from human videos for open-ended task generalization. *arXiv preprint*, 2026.

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*, 2025.

SUPPLEMENTARY MATERIAL

CONTENTS

A Implementation Details **16**

A.1 VLM-based architecture and visual targets 16

A.2 VGM-based architecture and visual targets 16

A.3 Attention permissions and backward supervision 16

A.4 Action-only deployment and optional visual prediction 17

A.5 Implementation of the future-conditioned WAM 17

B Simulation Training and Evaluation Configurations **18**

B.1 Training configurations 19

B.2 Default inference and evaluation protocol 20

C Real-World Training Hyperparameters **20**

C.1 VLM-based FDM 21

C.2 VGM-based FDM 21

C.3 Action-to-future ablation protocol 21

D Real-World Setup and Task Descriptions **22**

D.1 Drawer Packing 23

D.2 Box Packing and Closing 23

D.3 Shuttlecock Insertion 24

D.4 Paper-Cup Stacking 24

D.5 Block Stacking 25

E RoboTwin Per-Task Results **26**

A Implementation Details

Both instantiations connect action learning and future prediction through layer-wise attention between experts. Each expert maintains its own feature transformations, while attention allows it to read the permitted representations from other experts. The future-prediction branch therefore reads action-expert hidden states throughout the network, rather than the final numerical action output. Below, we describe the visual representations, attention permissions, and computation retained at deployment.

A.1 VLM-based architecture and visual targets

The VLM-based instantiation augments the understanding and action experts with a generation expert of the same architecture as the action expert. The generation expert is randomly initialized and has its own attention projections, feed-forward layers, and normalization parameters. For each camera view k, let $O_t^{(k)}$ denote its image within the current observation O_t. The generation expert receives visual context from $O_{t-15}^{(k)}$ and $O_t^{(k)}$, and predicts $O_{t+15}^{(k)}$, where offsets are measured in recorded frames. The understanding expert processes only the current images and task instruction. Thus, the additional temporal context supports future prediction during training without becoming an input requirement for the deployed policy.

A frozen Wan VAE encodes each image independently into a 48-channel latent grid with a spatial downsampling factor of 16. Independent encoding keeps the future target separate from the historical and current inputs. For the generation expert, a learned channel projection followed by a strided convolution converts the two context grids into a shorter sequence of visual tokens. After transformer processing, features at corresponding spatial locations are averaged across the two temporal positions separately for each view. A transposed convolution restores the latent-grid resolution, and normalization followed by a linear projection produces the predicted future latent. The visual objective averages latent reconstruction errors over valid camera views. This is direct regression: the generation expert predicts a clean latent in one forward pass, with a fixed zero time condition and no visual denoising loop.

A.2 VGM-based architecture and visual targets

The VGM-based instantiation pairs a generation expert with an action expert and connects corresponding transformer layers through attention. Their attention heads have matching dimensions, allowing action and video features to interact while retaining expert-specific transformations. Camera views are composed spatially before VAE encoding, so one video stream represents the multi-view scene. The generation expert processes both the current observation and future video; attention permissions distinguish their roles in action prediction.

During training, the current-frame latent is kept clean, while future-video latents and the demonstrated action chunk are corrupted using independently sampled noise and flow-matching times. The future-video tokens read the hidden representations of the noised action chunk at that training time. Video supervision is applied only to future latents, excluding the fixed current frame and masking padded temporal targets. The joint objective in Equation (7) uses $\lambda_a = \lambda_v = 1$ in both instantiations.

A.3 Attention permissions and backward supervision

Table 6 summarizes the attention permissions that implement the action-to-future connection. In the VLM-based model, understanding features are available to both downstream experts, and generation tokens additionally read action representations. In the VGM-based model, current-observation tokens attend only to the current observation within the visual–action sequence; action tokens read the current observation and other action tokens, while future-video tokens read all three groups. Future-video attention is bidirectional across future positions. Language conditioning is supplied separately to the VGM experts. These permissions hold at every layer, keeping both action representations and the observation features they read independent of generation-branch tokens.

The same connection carries supervision in the reverse direction during optimization. The future loss differentiates through the action keys and values read by the generation expert and updates the action representations that produce them. Action imitation simultaneously supervises those

Table 6: **Attention permissions between token groups.** Rows denote queries and columns denote keys/values; a checkmark permits attention. VLM generation tokens originate from historical and current image latents and are used to predict the future target. VGM language conditioning is separate from the groups shown here.

Model	Query group	Key/value group		
		Understanding	Action	Generation
VLM	Understanding	✓	✗	✗
	Action	✓	✓	✗
	Generation	✓	✓	✓
		Current observation	Action	Future video
VGM	Current observation	✓	✗	✗
	Action	✓	✓	✗
	Future video	✓	✓	✓

representations through the action velocity head. For the stop-gradient control in Section 4.3, the generation-facing action representations retain their forward values while their backward connection is blocked. Gradients through shared context are a separate path, so this control specifically isolates supervision through the action interface.

A.4 Action-only deployment and optional visual prediction

At deployment, the VLM-based policy uses its understanding and action experts to denoise an action chunk; the additional generation expert and its VAE processing are unnecessary for control. The VGM-based policy first processes the current observation and caches its keys and values at each transformer layer. Each action-denoising step reuses this cache, updating only the action stream without allocating or processing future-video tokens. The generation expert's observation processing is retained because it supplies the visual context for action generation.

Optional visual rollout uses the same action-to-future connection, with sampling schedules specific to each instantiation. Given a supplied action chunk, its representations condition future prediction with the action flow-matching time τ_a set to zero. For this optional rollout, the VLM-based implementation samples an action chunk before predicting future-image latents in a single pass. The VGM-based implementation can denoise actions and future video together using their respective schedules, restoring the fixed current-frame latent after each video update. Throughout this process, generation-branch tokens read action representations while the action stream remains independent of future predictions.

A.5 Implementation of the future-conditioned WAM

We implement the future-conditioned WAM used throughout the real-world task-success comparison (Section 4.2), the future-perturbation analysis (Section 4.4) and the efficiency comparison (Section 4.5) with a generation expert and an action expert connected through MoT attention. We retain the notation $C = (O_t, L)$ for the current observation and task instruction, A for the action chunk, and V for the clean future-video latents. This WAM realizes the future-to-action dependency in Table 1: action prediction reads the evolving future-video representations during joint denoising. The following description concerns this implementation of the WAM paradigm.

Visual and action representations. The generation expert uses the Wan video backbone, and a frozen Wan VAE encodes the spatially composed camera views into video latents. The current-frame latent provides the visual condition, while the subsequent latents represent V. Spatial patches of these latents form the generation expert's tokens. The action expert projects each vector in the noised action chunk A_{τ_a} into an action token and uses the action flow-matching time τ_a for time conditioning. Both experts receive text features of the task instruction L through their own cross-attention layers. Their transformer layers have matching attention-head dimensions and are connected at corresponding depths, while retaining separate projections, normalization layers, and feed-forward transformations.

An action velocity head produces $\widehat{u}_a$ from the action representations, and the generation expert predicts the future-video velocity.

Future-to-action attention. At every layer, action tokens attend to the current-observation tokens, future-video tokens, and other action tokens. The generation expert processes the visual sequence independently of action tokens. Within the visual sequence, current-observation tokens attend only to the current observation, while future-video tokens attend to the current observation and all future positions. With rows denoting queries and columns denoting keys/values in the order (current observation, action, future video), these permissions are

$$M_{\mathrm{WAM}} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \tag{8}$$

where each entry denotes an entire allowed or blocked attention block. Thus, action representations incorporate the current denoising estimate of the future through the generation expert's hidden states. Compared with the FDM-VGM permissions in Table 6, the cross-expert connection is reversed: WAM actions read future-video representations, whereas FDM future-video tokens read action representations.

Joint training. Training corrupts V according to Equation (5) and independently corrupts the demonstrated action chunk as

$$A_{\tau_a} = (1 - \tau_a)A + \tau_a \epsilon_a, \qquad \epsilon_a \sim \mathcal{N}(0, I). \tag{9}$$

The action and video flow-matching times, τ_a and τ_v, are sampled independently during training. The current-frame latent remains clean and receives a zero time condition. A joint forward pass predicts the action velocity $\widehat{u}_a$ and the future-video velocity, whose flow-matching targets are $\epsilon_a - A$ and $\epsilon_v - V$, respectively. The future-prediction loss $\mathcal{L}_{\mathrm{world}} = \mathcal{L}_{\mathrm{vid}}$ excludes the current frame and averages errors over valid future targets. The action loss $\mathcal{L}_{\mathrm{act}}$ similarly masks invalid action entries. For velocity supervision, it compares $\widehat{u}_a$ with $\epsilon_a - A$; for clean-action supervision, it compares $A_{\tau_a} - \tau_a \widehat{u}_a$ with A. Both forms train the same action velocity head used during sampling. The total objective combines the two losses with λ_a and λ_v as in Equation (7), with any time-dependent weighting applied within the respective loss. Through the permitted attention connection, action-loss gradients can update the visual representations read by the action expert.

Joint denoising at inference. Inference starts from Gaussian noise for both the action chunk and future-video latents, while fixing the current-frame latent to the encoded observation. The two streams use the same descending flow-matching times, $1 = \tau^{(0)} > \cdots > \tau^{(K)} = 0$, where K is the number of denoising steps. At step j, a joint forward pass evaluates both experts with $\tau_a = \tau_v = \tau^{(j)}$. Writing their velocity predictions as $\widehat{u}_a^{(j)}$ and $\widehat{u}_v^{(j)}$, the updates are

$$\begin{aligned} A_{\tau^{(j+1)}} &= A_{\tau^{(j)}} + \big(\tau^{(j+1)} - \tau^{(j)}\big)\widehat{u}_a^{(j)}, \\ V_{\tau^{(j+1)}} &= V_{\tau^{(j)}} + \big(\tau^{(j+1)} - \tau^{(j)}\big)\widehat{u}_v^{(j)}. \end{aligned} \tag{10}$$

The current-frame latent is restored after every video update. Consequently, future conditioning operates throughout sampling: the action expert reads partially denoised future representations at each step, without waiting for a completed video rollout. Returning only the action chunk omits video decoding, but still requires future-video latent computation during denoising. This dependence explains both why perturbations to future latents can change WAM actions in Section 4.4 and why future generation contributes to its action inference cost in Section 4.5.

B Simulation Training and Evaluation Configurations

We report the simulation training configurations for FDM-VLM and FDM-VGM on LIBERO and RoboTwin 2.0. Training settings and dataset splits are taken from the saved configurations and training logs. We use the following evaluation protocol for all reported experiments.

B.1 TRAINING CONFIGURATIONS

All four training runs use seed 42 and eight GPU processes, with 16 samples per device per microbatch. The effective global batch size additionally accounts for gradient accumulation, as listed in Table 7. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$ and $\epsilon = 10^{-8}$, gradient clipping at a norm of 1.0, BF16 precision, and DeepSpeed ZeRO-2. Gradient checkpointing is disabled in all runs. The learning rate follows cosine decay after warmup. Offline validation data are held out at the episode level using split seed 42; the episode counts in Table 7 follow the actual training logs. Offline validation measures held-out training objectives and is separate from closed-loop success-rate evaluation.

Table 7: Simulation training configurations. Counts and final optimizer steps follow the training logs. Validation intervals refer to offline validation.

	FDM-VLM		FDM-VGM	
Setting	LIBERO	RoboTwin	LIBERO	RoboTwin
Training episodes	1,659	26,950	1,659	27,225
Validation episodes	34	550	34	275
Validation fraction	2%	2%	2%	1%
Training steps	30,000	60,000	22,000	29,375
Warmup steps	1,000	1,000	1,100	1,468
Validation interval (steps)	2,000	2,000	200	500
Checkpoint interval (steps)	2,000	2,000	10,000	2,500
Per-device batch size	16	16	16	16
Gradient accumulation	1	2	1	8
Effective global batch size	128	256	128	1,024
Peak learning rate	2.5×10^{-5}	2.5×10^{-5}	10^{-4}	10^{-4}
Minimum learning rate	0	0	10^{-6}	10^{-6}
Weight decay	10^{-10}	10^{-10}	10^{-2}	10^{-2}
Camera views	2	3	2	3
Proprioceptive state input	False	False	False	False
Environment action dimension	7	14	7	14
Action prediction horizon	15	50	32	32
Normalization	Min–Max	Min–Max	Min–Max	Z-score

VLM-based FDM. The saved configurations initialize the policy from `pi05_base_pytorch`, with a Gemma-2B language backbone, a Gemma-300M action expert, and a So400m/14 vision encoder. The additional generation expert is randomly initialized and has not undergone pretraining. The maximum text length is 200 tokens, and action vectors are internally padded to 32 dimensions. LIBERO uses external and wrist camera views, while RoboTwin uses head, left-wrist, and right-wrist views, all at 224×224 resolution. Proprioceptive state input is disabled on both LIBERO and RoboTwin. Both datasets use Min–Max normalization. Compilation is enabled for these runs. The generation-expert configuration specifies the Wan VAE from Wan2.2, a resolution value of 256, a scale factor of four, and a loss weight of 1.0.

VGM-based FDM. The video backbone is initialized from Wan2.2-TI2V-5B. The action expert is randomly initialized and has not undergone pretraining; the initialization checkpoint loaded by the saved configurations stores these random initial weights. Both experts have 30 layers, with hidden dimensions of 3072 for the video backbone and 1024 for the action expert. Training uses cached UMT5 text features with a maximum text length of 128 tokens, and compilation is disabled. Action and future-prediction losses both have unit weight. Future frames are sampled every four steps, and the action prediction horizon is 32 for both benchmarks. The video and action schedulers each use 1,000 training timesteps and a shift of 5.0 for training and inference; action timesteps use the configured `beta_tau` sampling scheme. Proprioceptive state input is disabled on both benchmarks. LIBERO uses Min–Max normalization, while RoboTwin uses Z-score normalization.

Multi-view video inputs. For LIBERO, the VGM configuration uses two camera views, center-cropped and resized to 224×224 per view, then concatenated horizontally. For RoboTwin, the configured per-view size is 240×320. The current RoboTwin layout implementation assembles a 384×320 canvas, with a 256×320 head-camera view above two 128×160 wrist-camera views. These canvas dimensions are distinct from the configured per-view preprocessing size.

B.2 Default inference and evaluation protocol

The current evaluation defaults use the same environment protocol and action execution horizon for both model families within each benchmark (Table 8). The action prediction horizon and normalization remain specific to the model checkpoint. In particular, RoboTwin uses Min–Max normalization for FDM-VLM and Z-score normalization for FDM-VGM; evaluation-template fields do not override the normalization loaded by the policy server. The default sampling procedure uses 10 action denoising steps and evaluation seed 42.

Table 8: Current default simulation evaluation settings, shared by both model families except for the prediction horizon. Historical evaluation runs require verification against these defaults.

Setting	LIBERO	RoboTwin
Number of tasks	40	50
Episodes per task per condition	50	100
Evaluation seed	42	42
Action denoising steps	10	10
FDM-VLM action prediction horizon	15	50
FDM-VGM action prediction horizon	32	32
Action execution horizon	10	24

LIBERO. The default evaluation covers Spatial, Object, Goal, and LIBERO-Long (LIBERO-10), with 50 episodes for each of the 40 tasks, totaling 2,000 episodes per complete evaluation. Each episode starts from an official initial state and includes 30 initial settling steps. The control-step limits are 400 for Spatial, Object, and Goal, and 700 for Long, in addition to the settling steps. The policy receives the task description as its task instruction and outputs end-effector position and rotation increments with a binarized gripper command. Real-time chunking is disabled by default.

RoboTwin. The default evaluation covers all 50 tasks, with 100 episodes per task under each of the clean and randomized conditions, totaling 5,000 episodes per condition. Task environments determine initialization and episode limits. The policy uses absolute joint-position commands (`action_type=qpos`) and the unseen-instruction setting (`instruction_type=unseen`). By default, repeated observation acquisition is skipped while executing an action chunk. Per-task success rates are aggregated across the 50 tasks; the clean/randomized averaging convention is specified in Section E.

C Real-World Training Hyperparameters

We summarize the training configurations used for FDM-VLM and FDM-VGM in the real-world experiments. Both instantiations use the Wan VAE associated with the Wan2.2-5B backbone to encode visual inputs and prediction targets and to decode predicted latents. Both configurations use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$ and $\epsilon = 10^{-8}$, a gradient clipping norm of 1.0, and cosine learning-rate decay after 1,000 warmup steps. Training uses eight NVIDIA H100 GPUs with a per-device batch size of eight, giving an effective batch size of 64. Training uses DeepSpeed ZeRO-2, with gradient checkpointing disabled. The training budget is 30,000 optimizer steps. The action execution horizon is 25: both policies execute 25 actions from each predicted action chunk before replanning.

C.1 VLM-BASED FDM

For the $\pi_{0.5}$-based policy, we use a learning rate of 2.5×10^{-5}, decayed to zero, and weight decay of 10^{-10}. The policy takes three camera images at 224×224 resolution, with proprioceptive state input disabled. The action prediction horizon is 50 steps, and the model action dimension is 20. The action and future-prediction losses both have unit weight. Action denoising uses 10 steps. Table 9 lists the VLM-based settings.

Table 9: Training hyperparameters for FDM-VLM in the real-world experiments.

Hyperparameter	Value
Peak learning rate	2.5×10^{-5}
Minimum learning rate	0
Weight decay	10^{-10}
Training steps / warmup steps	30,000 / 1,000
Per-device / effective batch size	8 / 64
Camera views	3
Per-view image resolution	224×224
Proprioceptive state input	False
Action prediction horizon	50
Action execution horizon	25
Model action dimension	20
Action / future-prediction loss weight	1.0 / 1.0
Action denoising steps	10

C.2 VGM-BASED FDM

For the Wan-based policy, we use a learning rate of 10^{-4}, decayed to 10^{-6}, and weight decay of 10^{-2}. Each camera view is resized to 240×320 before multi-camera composition, and proprioceptive state input is disabled. The action prediction horizon is 32 steps, and future training frames are sampled with a stride of four. The model action dimension is 20. The action and future-prediction losses both have unit weight. Training uses cached text embeddings, and action denoising uses 10 steps. Table 10 lists the VGM-based settings.

Table 10: Training hyperparameters for FDM-VGM in the real-world experiments.

Hyperparameter	Value
Peak learning rate	10^{-4}
Minimum learning rate	10^{-6}
Weight decay	10^{-2}
Training steps / warmup steps	30,000 / 1,000
Per-device / effective batch size	8 / 64
Gradient accumulation steps	1
Per-view image resolution	240×320
Proprioceptive state input	False
Action prediction horizon	32
Action execution horizon	25
Model action dimension	20
Future-frame stride	4
Action / future-prediction loss weight	1.0 / 1.0
Action denoising steps	10

C.3 ACTION-TO-FUTURE ABLATION PROTOCOL

The ablation in Section 4.3 compares A. Action-only, B. Context-only future, C. Stop-gradient action conditioning, and D. Full FDM within each backbone family.

Matched training settings. Within each backbone and training seed, the four variants share the policy initialization, training data, action objective and weight, number of optimizer steps, effective batch size, and optimizer. B, C, and D additionally share the generation-expert architecture and initialization, future targets, and future-loss weight.

Stop-gradient implementation. In C, the stop-gradient operation applies to every action-representation interface read by the future-prediction branch, while leaving the action velocity head's access to H_a intact. All other attention permissions and shared-parameter update paths are held fixed between C and D. Gradients through shared context processing may remain in C; the comparison therefore isolates future-loss gradients through H_a. For the VGM-based FDM, all variants retain the current-observation computation required by the action expert.

Evaluation. All four variants use the same evaluation protocol and action-only inference on Drawer, Box, Shuttlecock, Cup, and Block (Section D). Table 4 reports per-task closed-loop success rates and their unweighted average separately for the VLM- and VGM-based instantiations, with comparisons using matched training seeds.

D Real-World Setup and Task Descriptions

We evaluate five manipulation tasks covering sequential object handling, stacking, cloth manipulation, and insertion. The descriptions below specify the task goals and constituent operations for the evaluation in Section 4.2.

Hardware and teleoperation. We use a dual-arm ARX-X5 robot and teleoperate it using a PICO 4 Ultra VR headset. The camera setup consists of one RealSense D435 providing a third-person view of the workspace and two RealSense D405 cameras, one mounted on each arm, providing left- and right-wrist views. Figure 6 illustrates the hardware setup.

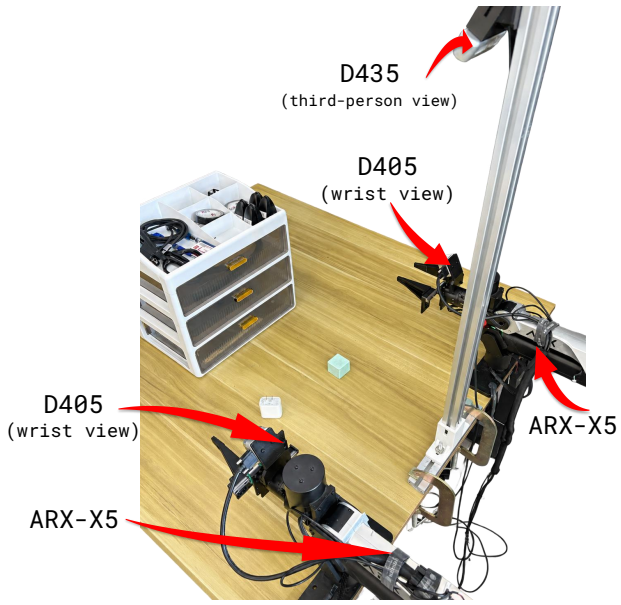


Figure 6: **Real-world experimental setup.** Two ARX-X5 robot arms are equipped with wrist-mounted D405 cameras, while a D435 camera provides a third-person view of the workspace.

Data collection and action parameterization. We collect 150 demonstration trajectories per task using VR teleoperation, reserving three trajectories for offline validation and using the remaining 147 for training. Incremental position and rotation commands from the VR interface are converted into robot joint commands through inverse kinematics. The control frequency is 30 Hz. Each arm uses a 10-dimensional action vector comprising three position components, a six-dimensional rotation

representation, and one gripper command, yielding a 20-dimensional bimanual action space. Actions are normalized using task-specific mean and standard deviation statistics.

Evaluation protocol. Real-world evaluation follows a blinded protocol: human evaluators are unaware of the identity of the model being evaluated. Task success is determined by a human evaluator based on whether the specified task goal is achieved. A trial is counted as a failure if it times out or a human evaluator judges that the robot cannot continue to complete the task. Object placements vary across demonstrations and evaluation trials, with slightly different placement distributions during training and evaluation. The task-specific placement variations and goals are described below.

D.1 Drawer Packing

The goal is to place both a charger and a block inside a drawer and leave the drawer closed (Figure 7). The drawer is initially closed, and both objects are outside the drawer in the workspace. The charger and block positions are randomized, while the drawer base position varies slightly. The robot first grasps the drawer handle and pulls outward to expose the interior. It then picks up each object, moves it over the open drawer, and releases it inside. Finally, the robot pushes the drawer back into the cabinet. This sequence couples articulated-object manipulation with pick-and-place: the opening must provide sufficient clearance for insertion, and the placed objects must not obstruct the drawer's closing motion. The desired final state has both objects contained in the drawer and the drawer fully closed.

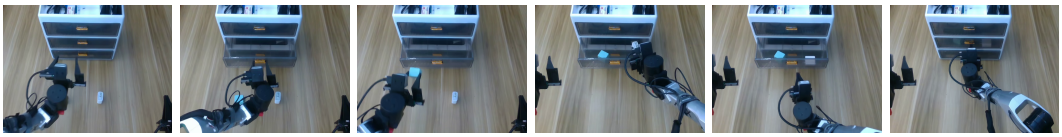


Figure 7: **Drawer packing.** The task requires opening the drawer, placing both the charger and block inside, and closing the drawer. The six frames are ordered chronologically from left to right.

D.2 Box Packing and Closing

The goal is to pack a scarf and a single flower into a box and close its lid (Figure 8). Initially, the box is open, with the scarf and flower lying on the tabletop beside it. The positions of the box, flower, and scarf vary slightly. The robot first lifts the scarf into the box and releases it within the box interior. It then grasps the flower by its stem, transfers it over the box, and places it on top of the scarf. After both objects have been deposited, the robot brings the hinged lid down over the opening. The task combines handling a deformable textile with positioning an elongated object and manipulating a lid. The contents must fit within the box so that neither the scarf nor the flower obstructs closure; the desired final state has both objects enclosed and the lid closed.



Figure 8: **Box packing and closing.** The robot places the scarf in the box, adds the flower, and closes the lid. The six frames are ordered chronologically from left to right.

D.3 SHUTTLECOCK INSERTION

The goal is to insert two shuttlecocks into a shuttlecock tube (Figure 9). The tube and the two shuttlecocks are initially separate on the tabletop. The tube position varies slightly, and the shuttlecocks are placed in front of the right arm with varying positions and orientations. One arm grasps and holds the tube horizontally, presenting its open end to the other arm. The other arm picks up a shuttlecock, aligns its narrow end with the opening, and advances it into the tube. The robot then repeats the pickup, alignment, and insertion sequence for the second shuttlecock while maintaining the tube's pose. This task requires coordination between the arm holding the receptacle and the arm performing insertion, as well as alignment of each shuttlecock with the tube axis. The desired final state has both shuttlecocks inside the tube.



Figure 9: **Shuttlecock insertion.** One arm holds the tube while the other inserts the two shuttlecocks sequentially. The six frames are ordered chronologically from left to right.

D.4 PAPER-CUP STACKING

The goal is to build a two-tier structure with three paper cups in the bottom row and two cups in the upper row (Figure 10). Initially, two cups are placed on the left and three on the right, with randomized positions within their respective sides. Starting from cups distributed across the tabletop, the robot picks up and arranges three cups side by side to form the base. It then lifts each of the remaining two cups and places it above a gap between adjacent base cups. Each upper cup must be supported by the cups beneath it, so the spacing of the base determines the available placement locations. The robot must control both the approach and release to avoid displacing the lightweight cups or disturbing an earlier placement. The desired final state is a freestanding five-cup structure that remains stable after the grippers withdraw.



Figure 10: **Paper-cup stacking.** The robot arranges three cups into a base and places two cups above them to form a two-tier stack. The six frames are ordered chronologically from left to right.

D.5 BLOCK STACKING

The goal is to stack two small blocks, red and yellow, on a larger blue base block (Figure 11). The blocks are initially separated on the tabletop. The two small blocks are randomly positioned, while the large central base block position varies slightly. In the illustrated sequence, the robot first picks up the red block, positions it over the blue block, and lowers it into contact before releasing it. It then picks up the yellow block and places it on top of the red block to complete the three-block stack. Each placement requires aligning the new block with the supporting block and withdrawing the gripper without disturbing the existing stack. The desired final state is a stable tower containing all three blocks in the prescribed order.



Figure 11: **Block stacking.** The robot places the small red block on the larger blue base block and then adds the small yellow block. The six frames are ordered chronologically from left to right.

E ROBOTWIN PER-TASK RESULTS

Table 11 lists the clean and randomized success rates for all 50 RoboTwin tasks for FDM-VLM and FDM-VGM. We compute each condition mean as the arithmetic mean of the 50 task success rates, and the overall mean as the arithmetic mean of the two condition means. The resulting overall success rates are 81.20% for the VLM-based FDM and 89.23% for the VGM-based FDM, as reported in Table 3.

Table 11: **Per-task success on RoboTwin 2.0.** Success rates are in percent. Each condition mean gives equal weight to all 50 tasks; the overall mean also weights clean and randomized conditions equally.

	FDM-VLM		FDM-VGM	
Task	Clean	Rand.	Clean	Rand.
adjust_bottle	100	100	100	100
beat_block_hammer	99	98	97	96
blocks_ranking_rgb	91	90	100	99
blocks_ranking_size	75	67	93	90
click_alarmclock	48	46	100	100
click_bell	94	90	100	100
dump_bin_bigbin	99	98	93	92
grab_roller	100	100	100	100
handover_block	64	76	42	60
handover_mic	100	100	96	98
hanging_mug	45	48	55	62
lift_pot	11	6	98	94
move_can_pot	55	65	91	99
move_pillbottle_pad	85	85	98	99
move_playingcard_away	99	96	99	100
move_stapler_pad	47	52	65	76
open_laptop	98	100	99	100
open_microwave	72	56	18	27
pick_diverse_bottles	70	69	77	85
pick_dual_bottles	75	74	89	93
place_a2b_left	92	95	94	94
place_a2b_right	94	89	93	98
place_bread_basket	87	81	92	97
place_bread_skillet	78	86	90	93
place_burger_fries	96	89	96	97
place_can_basket	64	58	71	60
place_cans_plasticbox	88	88	98	95
place_container_plate	100	97	96	99
place_dual_shoes	85	79	89	86
place_empty_cup	99	97	99	100
place_fan	94	85	96	99
place_mouse_pad	69	62	89	82
place_object_basket	61	74	82	80
place_object_scale	79	85	90	97
place_object_stand	87	90	94	92
place_phone_stand	97	95	99	100
place_shoe	95	97	96	97
press_stapler	100	99	93	93
put_bottles_dustbin	62	60	92	73
put_object_cabinet	80	80	89	89
rotate_qrcode	90	90	92	87
scan_object	71	62	93	82
shake_bottle	99	99	100	100
shake_bottle_horizontally	100	100	100	100
stack_blocks_three	87	82	95	96
stack_blocks_two	99	97	99	100
stack_bowls_three	74	83	73	82
stack_bowls_two	91	95	97	93
stamp_seal	63	65	85	87
turn_switch	69	68	60	63
Condition mean	81.54	80.86	88.84	89.62
Overall mean	81.20		89.23	