OPENWAM: AN OPEN, MODULAR EXPLORATION TOWARDS SYSTEMATIC WORLD–ACTION MODEL PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

World–Action Models (WAMs) inherit world knowledge from video-generative priors and channel it into executable control through embodied experience. Existing systems, however, are monolithic: backbone, visual representation, architecture, information flow, inference procedure, and training data are tightly coupled, obscuring which choices matter and why. We introduce `OpenWAM`, an open research stack that turns world–action pretraining into a controlled experimental program. `OpenWAM-Infra` factorizes the WAM design space into composable modules with unified training, inference, deployment, and evaluation. On this substrate, `OpenWAM-Study` asks what to inherit, how world and action learning interact, and how their synergy scales, and distills three principles: upstream knowledge transfers through a capable generative backbone and a compact, information-rich latent space; synergy requires dedicated action capacity, explicit world-to-action information flow, and synchronized joint denoising; and embodied pretraining principally improves out-of-domain generalization, with one-stage co-training over egocentric and robot data integrating world coverage and action grounding. Composing these principles, we build `OpenWAM-`$\boldsymbol{\alpha}$, an open WAM pretrained on 6,369 hours of egocentric human and robot data that delivers top-tier performance across eight simulation benchmarks and real-robot experiments spanning single-arm, bimanual, and dexterous-hand robots. We release the full stack: infrastructure, evaluation protocols, pretrained models, and data recipes. Website: `https://openwam-review.github.io`.

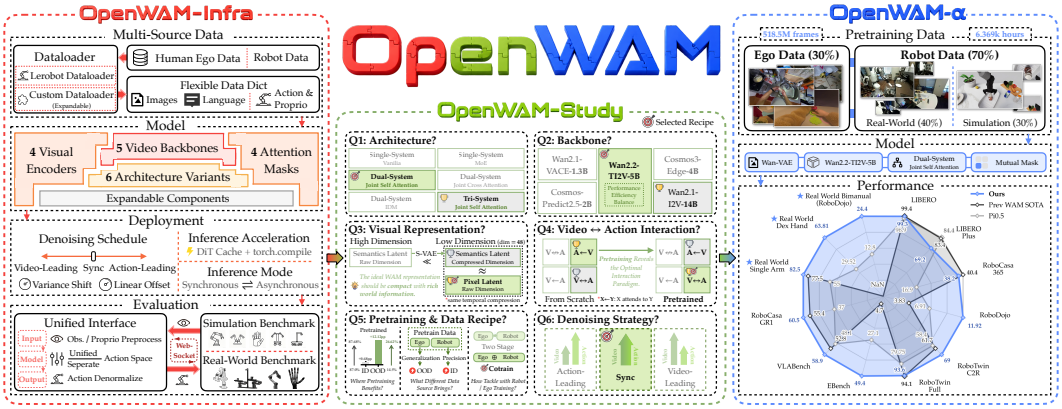

Figure 1: **Overview of `OpenWAM`. OpenWAM-Infra** (*left*) factorizes world–action modeling into composable modules with unified training, deployment, and evaluation; **OpenWAM-Study** (*middle*) resolves the design space through controlled questions and distills a pretraining recipe; **OpenWAM-**$\boldsymbol{\alpha}$ (*right*), pretrained on 518.5M frames of egocentric and robot data, sustains top-tier performance from simulation to the real world.

1 INTRODUCTION

Intelligence requires more than recognizing the world: an embodied agent must anticipate how the world changes and act to bring about desired changes. Vision–language models learn rich semantics

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.

from image–text data (Radford et al., 2021; Caron et al., 2021; Tong et al., 2024), and video generation models go further by synthesizing how visual worlds evolve (Ho et al., 2022; Brooks et al., 2024); an embodied system, however, must learn not only *what can happen*, but also *what actions make it happen*. This exposes a fundamental asymmetry: videos of the changing world are abundant, whereas robot trajectories with action labels remain scarce (Ye et al., 2026d). Whereas earlier policies such as ACT (Zhao et al., 2023) and Diffusion Policy (Chi et al., 2025) learn solely from such demonstrations and Vision–Language–Action (VLA) models (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024) inherit semantic knowledge from vision–language pretraining, World–Action Models (WAMs) (Pai et al., 2025; Ye et al., 2026c; Li et al., 2026b) *inherit world knowledge from video generation priors, then learn how to act through embodied experience*.

The promise of a WAM lies not in attaching an action head to a video generator but in learning world prediction and action generation in *synergy*: world modeling supplies structured knowledge of states, dynamics, and possible futures, while action learning focuses the model on the changes that matter for control. Realizing this synergy is nontrivial: useful knowledge may reside in different parts of an upstream video model, joint prediction may lack an effective information path, and a design that fits one domain may not transfer to new scenes or embodiments. This leads to three questions:

> *Question 1:* **What world knowledge should a WAM inherit?**
> *Question 2:* **How can we create synergy between world and action learning?**
> *Question 3:* **How can this synergy be scaled across domains?**

Answering these questions is difficult with existing monolithic systems, whose generative backbone, visual representation, architecture, information flow, inference procedure, and training data are tightly coupled (Kim et al., 2026b; Bi et al., 2026; Zhang et al., 2026b). We therefore introduce **OpenWAM**, an open research stack for systematically developing WAMs. **OpenWAM-Infra** factorizes the WAM design space into modular components with unified training, inference, deployment, and evaluation across domains and embodiments (Section 2), turning world–action modeling into a controlled experimental program.

On this substrate, **OpenWAM-Study** investigates the design principles of WAMs through controlled experiments (Section 3) and produces three findings: upstream world knowledge transfers most effectively through a capable generative backbone and a compact, information-rich visual latent space; world–action synergy does not emerge from parameter count or joint prediction alone but requires action-specific capacity, explicit world-to-action information flow, and a joint test-time denoising schedule; and embodied pretraining primarily improves out-of-domain generalization rather than in-domain fitting, with egocentric human video broadening world coverage, robot trajectories grounding executable actions, and their joint training the strongest integration strategy. Together, these findings form a recipe from *inheriting* world knowledge, to *enabling* world–action synergy, to *scaling* it across domains.

Finally, we compose these principles into **OpenWAM-α**, an open pretrained WAM (Section 4). Pretrained on 518.5M frames (6,369 hours) of egocentric human and robot data through a unified 80-D action space, OpenWAM-α shows that the recipe remains effective at scale: it delivers consistently strong results on eight simulation benchmarks spanning five embodiment categories and on single-arm, bimanual, and dexterous-hand real robots, while these large-scale evaluations yield further insights into the design and scaling of embodied foundation models.

In summary, our contributions are:

- **OpenWAM-Infra** (Section 2): a modular infrastructure for controlled WAM research.
- **OpenWAM-Study** (Section 3): a controlled study distilling three principles: inheriting upstream world knowledge, enabling world–action synergy, and scaling it across domains.
- **OpenWAM-α** (Section 4): an open pretrained WAM scaling these principles, released with the full stack (infrastructure, evaluation protocols, weights, and data recipes).

We provide a detailed discussion of related work in Appendix B.

2 OpenWAM-Infra: A Modular Infrastructure for World–Action Modeling

OpenWAM-Infra factors world–action modeling into four components with explicit interfaces — a *composable model*, a *training runtime*, a *deployment runtime*, and an *evaluation protocol* — related only by one-way dependencies (Figure 2). This decoupling keeps the codebase extensible and provides the substrate on which **OpenWAM-Study** (Section 3) runs controlled experiments and **OpenWAM-α** (Section 4) is instantiated at scale; code contracts, module variants, and formulations are specified in Appendix C.

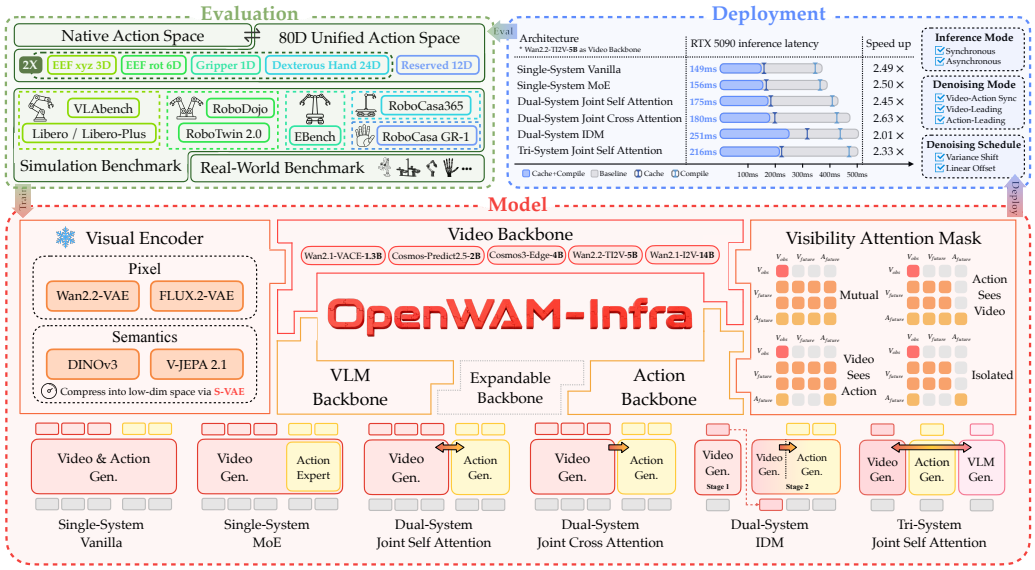

Figure 2: **Overview of OpenWAM-Infra.** *Model* (bottom): visual encoder $\mathcal{E}$, stream backbones $\mathcal{S}$, and visibility mask $\mathcal{M}$, composed into six architectures. *Deployment* (top right): inference modes, denoising schedules, and latency (RTX 5090). *Evaluation* (top left): benchmarks and action spaces.

Composable Model. A composition rule C assembles three classes of interchangeable modules into an architecture $C(\mathcal{E}, \mathcal{S}, \mathcal{M})$. The frozen **visual encoder** $\mathcal{E}$ maps observations into the latents the world stream predicts; OpenWAM-Infra ships two reconstructive encoders (Wan2.2-VAE, FLUX.2-VAE), two representation encoders (DINOv3, V-JEPA 2.1), and an optional S-VAE adapter that compresses the latent dimension (Wan et al., 2025; Black Forest Labs, 2025; Siméoni et al., 2025; Mur-Labadia et al., 2026; Zhang et al., 2025a). The **stream backbones** $\mathcal{S}$ execute the world, action, and optional understanding streams: five video backbones from 1.3B to 14B parameters (Wan et al., 2025; Ali et al., 2025), the Qwen3-VL family (Bai et al., 2025) as the VLM backbone, and an action backbone either dedicated (ActionDiT) or shared with the video backbone. The **visibility attention mask** $\mathcal{M}$ governs which tokens may attend across streams in four modes (*mutual*, *action-sees-video*, *video-sees-action*, and *isolated*). Composition yields six architectures in three families (Figure 2, bottom): **Single-System** (*Vanilla*, *MoE*), **Dual-System** (*joint self-attention*, *joint cross-attention*, *IDM*), and **Tri-System** (*joint self-attention*), all assembled from configuration.

Training Runtime. One trainer trains every architecture against one sample contract and one joint flow-matching objective without inspecting architecture internals. The video and action streams are noised to *independently* sampled timesteps (t_v, t_a), so training covers the entire joint noise plane and any inference schedule stays in-distribution. Training scales through DeepSpeed ZeRO, mixed precision, and gradient checkpointing across pretraining, fine-tuning, and exact-resume workflows, and every run emits a **self-contained checkpoint** (weights, resolved module specification, and action-normalization statistics) from which fine-tuning and deployment rebuild the architecture.

Deployment Runtime. One policy server serves every checkpoint and keeps two choices orthogonal: *when* inference runs and *how* the two streams are denoised. *Synchronous* and *asynchronous*

inference modes (the latter prefetching the next action chunk) share one action buffer and are indistinguishable to the client. Three **denoising schedules** trace paths through the joint noise plane: *sync* along the diagonal, and the *variance-shift* and *linear-offset* families that let either stream lead, with synchronous serving recovered as a special case. Four independently configurable **accelerations** (prompt-embedding cache, video-decode skip, `torch.compile` under CUDA graphs, and a DiT velocity cache) yield 2.0–2.6× serving speedups across the six architectures (Figure 2, top right).

Evaluation Protocol. Benchmarks connect to the policy server as thin WebSocket clients that import nothing from the model or training stack: a request carries camera views, the instruction, and optionally the robot state, and returns one action in native units. **Eight simulation benchmarks** are integrated (LIBERO, LIBERO-Plus, VLABench, RoboTwin2.0, RoboDojo, RoboCasa365, RoboCasa-GR1, and EBench) (Liu et al., 2023; Fei et al., 2025; Zhang et al., 2024; Chen et al., 2025; 2026b; Nasiriany et al., 2026; NVIDIA et al., 2025; Nasiriany et al., 2024; Gao et al., 2026), and real robots connect through the same interface. Actions cross the pipeline in each benchmark's **native action space** or in an **80-D unified action space** with fixed slot semantics, so that one action head spans heterogeneous embodiments.

3 OpenWAM-Study: Design Principles for World–Action Models

Building on **OpenWAM-Infra**, we analyze the design principles of WAMs through controlled experiments, from which upstream priors to inherit (Section 3.1), to how world and action learning should interact (Section 3.2), to which choices survive cross-domain pretraining (Section 3.3). All experiments use **RoboTwin2.0** (Chen et al., 2025), a bimanual benchmark of over 50 tasks, under two protocols: **RoboTwin2.0-Full** (Bi et al., 2026) trains on 2,500 clean and 25,000 scene-randomized demonstrations and evaluates on both; **RoboTwin2.0-Clean2Random** (Yuan et al., 2026a) trains on clean data only, so clean evaluation is in-domain (ID) and randomized evaluation is out-of-domain (OOD). Success rate is the metric throughout; complete experimental descriptions are provided in Appendix D.

3.1 Inheriting Upstream World Knowledge

Question 1: What world knowledge should WAMs inherit, and how is it best inherited?

Generative World Priors. We compare four video backbones of increasing size, Wan2.1-VACE-1.3B, Cosmos-Predict2.5-2B, Wan2.2-TI2V-5B, and Wan2.1-I2V-14B (Wan et al., 2025; Ali et al., 2025), each instantiated as a dual-system WAM, on RoboTwin2.0-Full. Success rate improves steadily with backbone capacity (Figure 3): Wan2.1-I2V-14B performs best, yet Wan2.2-TI2V-5B trails it by only 1.40 points at about a third of the parameters, so we adopt the 5B model as the default. Since the backbones also differ in architecture, pretraining data, and objective, backbone choice is a consequential design variable, though the gain cannot be attributed to size alone.

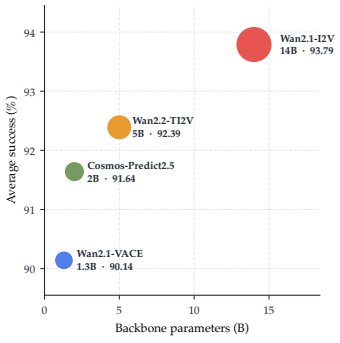

Figure 3: **Backbone size.** WAM performance improves consistently with video backbone capacity.

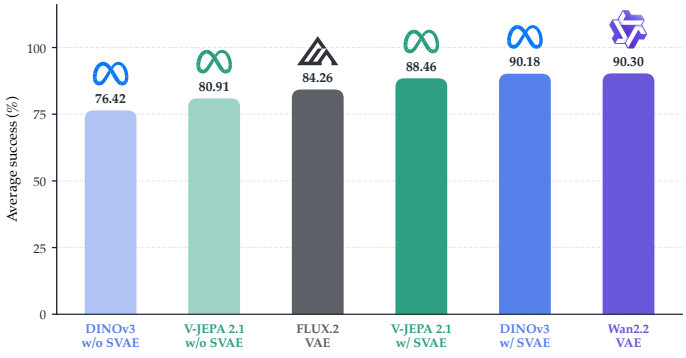

Figure 4: **Visual representation priors.** WAMs built on reconstructive and representation encoders, the latter also with an S-VAE (Zhang et al., 2025a) adapter that contracts high-dimensional latents for DiT processing.

Visual Representation Priors. Vision encoders are either *reconstructive*, trained for pixel reconstruction, or *representation* encoders, trained to yield semantically structured features. Motivated by generative modeling on representation latents (Zheng et al., 2026a; Jha et al., 2026), we compare Wan2.2-VAE (Wan et al., 2025) and FLUX.2-VAE (Black Forest Labs, 2025) against DINOv3 (Siméoni et al., 2025) and V-JEPA 2.1 (Mur-Labadia et al., 2026). To isolate the representation, every model keeps the Wan2.2-TI2V-5B architecture with a randomly initialized DiT, and the native $4\times$ temporal compression of Wan2.2-VAE is imposed on the other encoders by averaging every four frames. Because DiTs are tuned to low-dimensional latents, naive use of the 768-D (DINOv3) or 1024-D (V-JEPA 2.1) features underperforms, so following Jha et al. (2026) we also train an S-VAE (Zhang et al., 2025a) adapter contracting them to 48-D. As Figure 4 shows, the naive representation variants fall below FLUX.2-VAE, but their contracted variants significantly outperform it, and DINOv3 w/ S-VAE nearly matches Wan2.2-VAE, whose remaining margin we attribute to its native temporal compression and co-design with the backbone. What matters is thus not the reconstructive-versus-representation divide but the latents' operational properties, temporal and token compactness with rich world information; representation priors can thus be inherited through temporal compression and dimension contraction (Appendix D.3).

Finding 1: A WAM inherits upstream world knowledge most effectively through a capable generative backbone and a compact, information-rich representation space. Reconstructive encoders are not the only option; representation encoders with dimension compression are also performant.

3.2 Building Synergy between World and Action Learning

Inheriting the right priors is not enough. A WAM must also build synergy between world and action learning, which requires three decisions: where action-specific capacity lives, which cross-stream information paths are available during training, and whether inference preserves the noise-state relationship under which those paths were learned.

Question 2: How should inherited world knowledge interact with action learning?

Table 1: **Architecture Ablation.** Averaged success rates (%) on RoboTwin2.0-Full. Bold denotes best values.

Architecture		Success Rate (%)		
System	Variant	Clean	Randomized	Average
Single-System	**Vanilla**	85.20	85.80	85.50
	MoE	86.22	83.04	84.63
Dual-System	**Joint Self-Attention**	92.34	**92.38**	92.36
	Joint Cross-Attention	87.86	88.64	88.25
	Detached Cross-Attention	92.06	91.64	91.85
	IDM	87.76	88.14	87.95
Tri-System	**Joint Self-Attention**	**92.84**	92.36	**92.60**

Table 2: **Action learning requires access to world information.** Success rates (%) on RoboTwin2.0-Full.

Mask	**Success Rate (%)**		
	Clean	Random.	Average
Isolated	88.08	86.74	87.41
Video Sees Action	87.92	87.34	87.63
Action Sees Video	**92.98**	**91.80**	**92.39**
Mutual	92.50	**91.80**	92.15

Architectural Capacity. The three architecture families of Section 2 span the axis from shared to dedicated action capacity. We evaluate all six variants, with joint cross-attention both end-to-end and with gradients detached at the video features, for seven models in total (Table 1). Performance improves steadily from single- to dual- to tri-system: joint self-attention is the strongest dual-system variant (92.36% on average), and the tri-system model is best overall (92.60%). Balancing performance against complexity and isolating the world–action interaction from the influence of the VLM's understanding features, we adopt dual-system joint self-attention for the remaining experiments, so that the subsequent findings reflect this interaction alone.

Training-Time Information Flow. Joint attention fixes the interface between the streams but not which information flow creates synergy, so we compare the four attention masks of Section 2. The results split by whether the action stream can access world features (Table 2): isolated and video-sees-action masks trail by about five points, whereas action-sees-video and mutual visibility retain strong performance (92.39% and 92.15%). World-to-action flow is therefore necessary, while adding the reverse action-to-world path changes the from-scratch result only marginally, leaving one-way and mutual visibility as two viable masks that we revisit after pretraining.

Inference-Time Information Flow. Training-time masking gates cross-stream flow *fully*, whereas the inference-time schedule gates it only *partially*. Fixing mutual visibility, we compare synchronized denoising with variance-shift and linear-offset schedules letting either stream lead (Baade et al., 2026), sweeping α and o in both directions (Figure 5). Synchronized denoising performs best and no asynchronous schedule improves on it; among them, video-leading is better under variance shift and action-leading under linear offset. Explicit video-to-action flow is thus necessary in training but yields no gain when enforced at inference, supporting the representation-learning hypothesis of world–action modeling (Yuan et al., 2026b) over implicit planning-then-IDM (Ye et al., 2026c) (Appendix D.6).

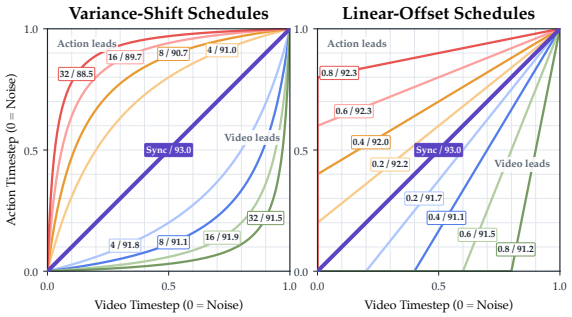

Figure 5: **Denoising schedules.** Curves above/below the diagonal denoise action/video first.

Finding 2: World–action synergy requires explicit world-to-action information flow in training, and synchronized denoising at inference. We carry forward dual-system joint self-attention with synchronized denoising and defer the choice between one-way and mutual visibility to pretraining.

3.3 Consolidating Knowledge Across Domains

Strong single-domain performance alone does not establish transferable world–action knowledge, since a model may simply fit the visual and action distribution of the target tasks. Robot trajectories provide action supervision but limited visual coverage; egocentric video offers the reverse. We therefore ask where pretraining gains arise, how the sources should be combined, and whether the information-flow choice made from scratch survives pretraining.

Question 3: How can world–action knowledge be consolidated and transferred across domains?

Setup. All runs keep the backbones, optimization budget, and inference procedure fixed, using the architecture and denoising schedule selected above, and vary only whether and how the model is pretrained with embodiment data. Under an identical 600-hour budget, we compare fine-tuning from scratch with three pretraining strategies: *robot-only* spends the full budget on real-robot trajectories from RoboCOIN (Wu et al., 2025), while two mixed variants combine 350 hours of egocentric video from EgoDex (Hoque et al., 2025) with 250 hours of robot data, either in two stages (*ego then robot*) or jointly (*ego + robot co-train*). All variants undergo identical downstream fine-tuning under both protocols (Appendix D.7).

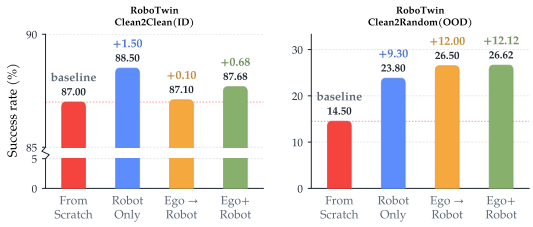

Figure 6: **Embodied pretraining primarily improves OOD generalization.** Success rates on Clean2Random (ID and OOD).

Results. Embodied pretraining yields modest in-domain gains but strong OOD gains (Figure 6); it therefore contributes knowledge that transfers beyond the downstream distribution rather than better fitting a saturated ID benchmark. The two sources contribute different strengths: robot-only pretraining gives the strongest ID performance, whereas both mixed strategies generalize better OOD, consistent with robot trajectories strengthening action grounding and egocentric video broadening the visual and interaction dis-

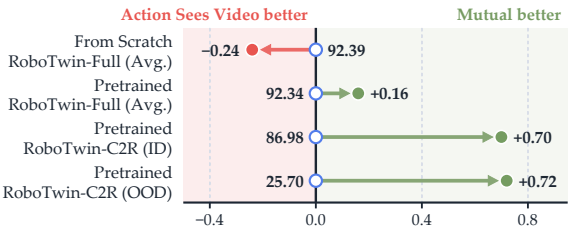

Figure 7: **Pretraining changes the preferred information flow.** Arrows run from Action Sees Video to the matched Mutual result (green: gain, red: drop).

tribution. Sequential training and one-stage co-training are nearly matched, so using both sources matters more than their ordering; co-training is marginally strongest and removes the curriculum transition, so we adopt it as the default. Pretraining also changes the preferred information flow: RoboTwin2.0-Full slightly favors one-way visibility from scratch but mutual visibility after pretraining, and RoboTwin2.0-Clean2Random shows the same reversal in both ID and OOD (Figure 7). The reversal is not an artifact of domain shift or protocol: with abundant pretraining data the interaction becomes bidirectional, predicted frames guiding action generation while predicted actions inform the manipulator's motion, whereas from-scratch data scarcity prevents the streams from establishing such correspondences. We therefore carry mutual visibility forward.

Finding 3: Embodied pretraining primarily expands OOD generalization. Robot trajectories preserve action grounding, egocentric video broadens transfer, and one-stage co-training integrates both effectively. At pretrained scale, mutual visibility is consistently preferred.

4 OpenWAM-α: From Principles to a Pretrained Model

We instantiate the findings of **OpenWAM-Study** at scale in **OpenWAM-α**, an open foundation world–action model pretrained on 518.5M frames (6,369 hours) of egocentric human and robot data. Below we specify its architecture, training, and deployment (Section 4.1) and evaluate it on eight simulation benchmarks (Section 4.2) and three real-robot embodiments (Section 4.3); the data mixture, curation pipeline, full model specification, and per-task results are given in Appendix E.

4.1 Architecture, Training, and Deployment

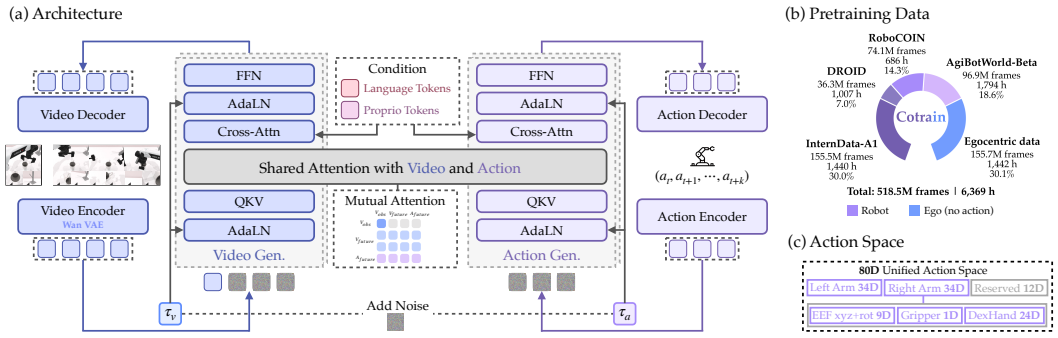

Figure 8: **Overview of OpenWAM-α.** (a) A video-generation DiT and an ActionDiT jointly denoise the future frames and the action chunk through shared attention under the mutual visibility mask. (b) The one-stage ego + robot pretraining mixture. (c) The 80-D unified action space.

Architecture. OpenWAM-α adopts the architecture that **OpenWAM-Study** converges to (Section 3): the frozen Wan2.2-VAE as visual encoder, the pretrained Wan2.2-TI2V-5B DiT (Wan et al., 2025) as world stream and a dedicated 1B-parameter ActionDiT as action stream, bridged at all 30 paired layers through joint self-attention under the *mutual* visibility mask with first-frame-causal intra-video attention (Figure 8a). Conditioned on the current observation, the language instruction, and the proprioceptive state in the 80-D unified action space, the model jointly denoises the future frames of a T-frame window in latent space together with a continuous action chunk (Zhao et al., 2023); both streams read the conditions through per-block cross-attention.

Training. Following the one-stage co-training recipe of Section 3.3, OpenWAM-α is trained end to end on a five-source ego + robot mixture (Figure 8b): egocentric human video (30% of frames), real-world robot data from 17 platforms (40%), and synthetic robot data (30%), each cleaned by a two-level curation pipeline (Appendix E.4). Embodiments meet in the unified action space (Figure 8c), where a validity mask confines action supervision to populated coordinates, so egocentric data supervises only the world stream. The DiT, ActionDiT, and proprioception encoder are trained under the joint flow-matching objective with independently sampled per-stream timesteps; the text encoder and VAE remain frozen.

Deployment. At test time, OpenWAM-α follows the synchronized schedule of Section 3.2: both streams advance in lockstep over $N = 10$ denoising steps, the first latent frame stays clamped to the

current observation, and the finished chunk is mapped back to each robot's native action space. All evaluations use synchronous inference, so results reflect the model's own predictions; served through the **OpenWAM-Infra** acceleration stack, one chunk takes roughly 170 ms on an RTX 5090.

4.2 Simulation Benchmark Evaluation

From the pretrained model, we fine-tune and evaluate OpenWAM-α on the eight simulation benchmarks of Section 2; RoboTwin2.0 is evaluated under both the Full and Clean2Random protocols. Figure 9 compares OpenWAM-α with representative VLA and WAM baselines and the best of each family (details in Appendices F.2 and H). We ask two questions: **how does OpenWAM-α perform, and between VLA and WAM, which paradigm prevails?**

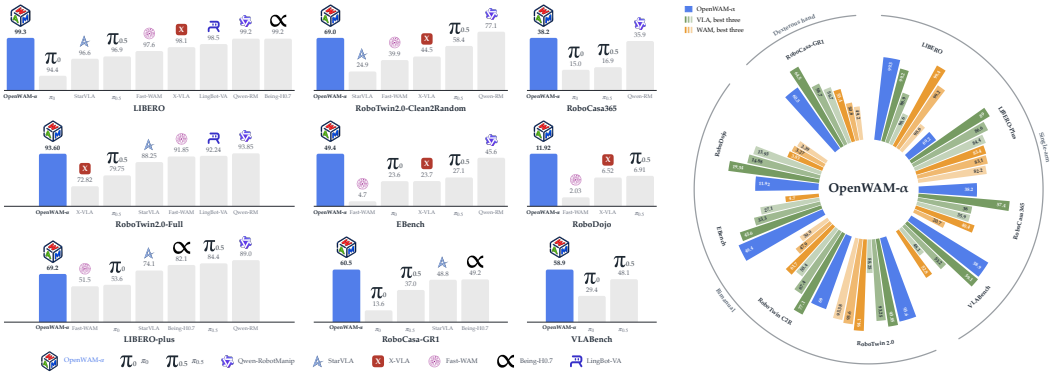

Figure 9: **Simulation benchmark results of OpenWAM-α.** *Left*: scores against representative VLA and WAM baselines on every benchmark. *Right*: OpenWAM-α against the best three VLAs and WAMs per benchmark, grouped by embodiment.

How Does OpenWAM-α Perform? Across most benchmarks, OpenWAM-α sits in the top tier: within a marginal gap of the best model on LIBERO, VLABench, RoboTwin2.0-Full, RoboCasa365, and RoboCasa-GR1; state of the art on EBench, leading the runner-up QWEN-ROBOTMANIP by roughly 4 points; and the strongest WAM by a clear margin on RoboTwin2.0-Clean2Random and RoboDojo, where the best models are VLAs. The one exception is LIBERO-Plus, where its scores fall markedly below its standing elsewhere, with the losses concentrated under the *camera* and *noise* perturbations. Comparing against the WAMs that perform strongly there (ABot-M0.5, ImageWAM, Being-H0.7) traces the deficit to two causes (Appendix E.5): far less single-arm pretraining data, and a pixel-level future prediction that is inherently sensitive to camera and noise disturbance.

Takeaway 1: How well an embodied model generalizes on a benchmark is determined by whether its pretraining mixture contains data close to the benchmark's test conditions; extending Section 3.3, large-scale, diverse robot data remains the decisive ingredient of an embodied foundation model.

Takeaway 2: With large-scale data, WAMs that predict the future in a pixel latent space achieve excellent performance, yet their robustness to visual disturbance is inherently limited; echoing Section 3.1, a compact, information-rich representation robust to environmental variation is still needed.

VLA versus WAM: Which Paradigm Prevails? Across all benchmarks (Figure 9, right), the VLA and WAM groups show no substantial gap in overall success rate, and each has standout models atop leaderboards: ABot-M0.5 and OpenWAM-α among WAMs, Xiaomi-Robotics-1 and QWEN-ROBOTMANIP among VLAs. Since these models differ in data, architecture, and training, neither paradigm is superior outright, yet the fine-grained scores reveal a consistent pattern (Figure 10). *In distribution, WAMs fit better.* Between StarVLA and Fast-WAM, two models without embodied pretraining, the WAM scores visibly higher on LIBERO, RoboTwin2.0-Full, and the Clean split of RoboTwin2.0-Clean2Random, and OpenWAM-α likewise leads the three strongest VLAs on LIBERO, VLABench (in-distribution), and RoboTwin2.0-Clean2Random (Clean) despite their larger pretraining data. This advantage traces to video-latent supervision, which injects information beyond action supervision alone and lets a WAM fit the training data more closely. *Out of distribution, VLAs generalize better.* Both comparisons reverse on LIBERO-Plus, the Randomized split of RoboTwin2.0-Clean2Random, and the Open split of RoboDojo, where the WAMs trail by

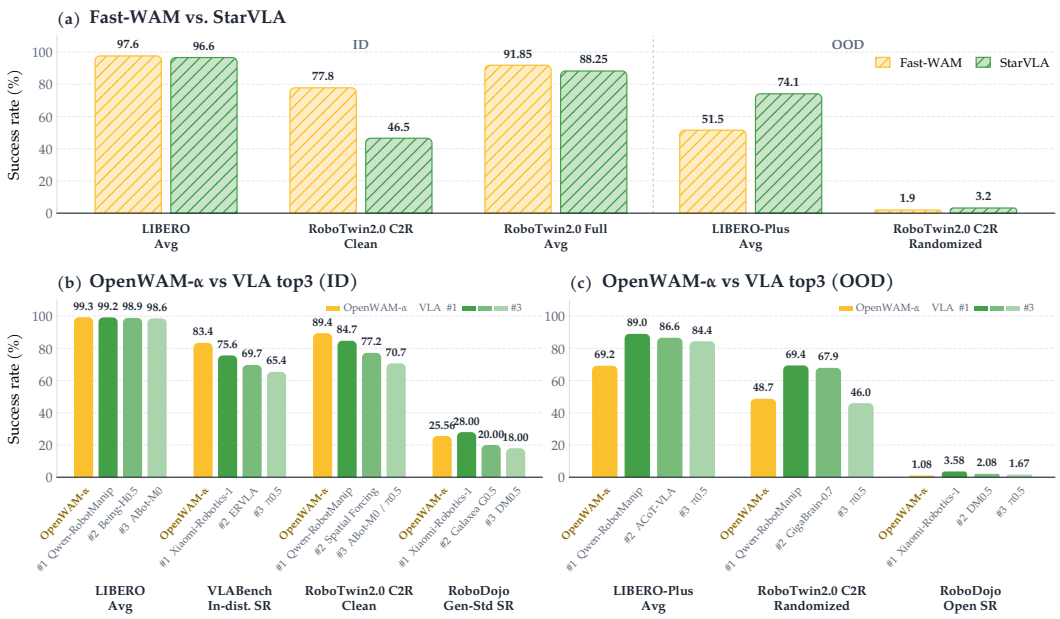

Figure 10: **ID and OOD comparisons between the two paradigms.** (a) Fast-WAM versus StarVLA, two models without embodied pretraining, on ID and OOD splits. (b, c) OpenWAM-α versus the three strongest VLAs on ID and OOD splits.

an evident margin: the long-horizon future prediction that helps fitting also accumulates error under distribution shift, a burden the action-only VLA never carries. Crucially, both deficits are remediable by data: sufficiently rich pretraining data with precise action annotation lets either paradigm achieve strong fitting and generalization, so data remains the first priority.

Takeaway 3: Neither paradigm prevails outright: video-latent supervision gives WAMs the edge in in-distribution fitting, VLAs generalize better out of distribution, and either deficit can be compensated by large, diverse pretraining data; combining their strengths is a promising route.

4.3 Real-Robot Evaluation

Beyond simulation, we evaluate OpenWAM-α on three real-robot embodiments: single-arm stacking, pick-and-place, and hanging tasks on a Franka-Research-3; the bimanual RoboDojo real-robot benchmark (ARX X5, Piper, and Piper X) covering five task categories; and four dexterous-hand tasks on a Wuji hand with a Tianji arm, an embodiment whose 30-D action space never appears in pretraining (details in Appendices E.6 and G). OpenWAM-α leads LingBot-VA (a WAM) and $\pi_{0.5}$ (a VLA) on most single-arm tasks and in average success rate, tops the RoboDojo-Real leaderboard with state-of-the-art results on most tasks across the three embodiments, and outperforms $\pi_{0.5}$ by a clear margin on all dexterous-hand tasks under both in-domain and out-of-domain setups. It thus stands on par with today's leading models across all three embodiments, a strong baseline for the broader research community.

5 Conclusions

This work introduced **OpenWAM**, an open research stack that turns world–action modeling from tightly coupled implementation choices into a controlled experimental program. **OpenWAM-Infra** factorizes the WAM design space into composable modules served by a single trainer, policy server, and evaluation protocol; **OpenWAM-Study** examined what world knowledge a WAM should inherit, how world and action learning create synergy, and how that synergy consolidates across domains; and **OpenWAM-α** instantiated this recipe at scale on egocentric human and robot data, delivering consistently strong results in simulation and on single-arm, bimanual, and dexterous-hand real robots. We release the full stack (infrastructure, evaluation protocols, weights, and data recipes); limitations and future directions are discussed in Appendix A.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code development, to polish the writing, and to produce some of the figures. We did not use generative AI tools to develop the research ideas or methodology, to design or interpret the experiments, or to outline the paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human-subject studies. The egocentric human data and the robot data used for pretraining were obtained through standard commercial and public channels in full compliance with their respective terms and licenses, and all benchmarks are publicly available. Real-robot experiments were conducted in controlled laboratory settings under human supervision with standard safety measures. We release code, pretrained weights, and data recipes to support reproducible research; the released models are general-purpose manipulation policies and are not intended for applications that could cause harm to people. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

Code, pretrained weights, and data recipes are available in an anonymous repository at `https://github.com/openwam-review/OpenWAM`, including the **OpenWAM-Infra** codebase (composable models, trainer, policy server, and benchmark clients) and the scripts for the experiments in this paper. The complete specification of OpenWAM-$\boldsymbol{\alpha}$ is given in Appendices E.1 to E.3, the pretraining data mixture and curation pipeline in Appendix E.4, the pretraining hyperparameters in Appendix F.1, and the per-benchmark SFT configurations in Appendix F.2. The protocols and full results of the controlled studies are described in Appendix D, the real-robot task setups and evaluation protocols in Appendix G, and the complete per-benchmark simulation results in Appendix H.

REFERENCES

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 arXiv:2511.00062*, 2025.

Zeyuan Allen-Zhu. Physics of language models: Part 4.1, architecture design and the magic of canon layers. *Advances in Neural Information Processing Systems*, 38:42349–42369, 2026.

Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation. *arXiv preprint arXiv:2602.11401*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL `https://arxiv.org/abs/2511.21631`.

Yongjie Bai, Hanting Wang, Mingtong Dai, Qijun Zhong, Yang Liu, and Liang Lin. Bridge-wa: Predicting where and how the world changes for robotic action. *arXiv preprint arXiv:2607.02195*, 2026.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11 (113):eaea6201, 2026.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

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

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

Black Forest Labs. FLUX.2: Analyzing and enhancing the latent space of FLUX. Technical blog, 2025. URL https://bfl.ai/research/representation-comparison.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, Shu Jiang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, et al. Aha-wam: Asynchronous horizon-adaptive world-action modeling with observation-guided context routing. *arXiv preprint arXiv:2606.09811*, 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 arXiv:2601.02456*, 2026b.

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026c.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, Haoning Wu, Haoyun Liu, Yuzhi Chen, Lulu Zheng, Botai Yuan, et al. Abot-m0.5: Unified mobility-and-manipulation world action model. *arXiv preprint arXiv:2607.00678*, 2026a.

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

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, Honghao Su, Zhiyang Dou, Kaixuan Wang, Dandan Zhang, Yunze Liu, Yan Qin, Qiwei Liang, Qiwei Wu, Zijian Lin, Wenwei Lin, Yuran Wang, Minghua He, Tianshu Wu, Ruihai Wu, Jingquan Zhou, Kai-Chong Lei, Haibao Yu, Yuanfeng Ji, Weiyang Jin, Guanyu Lin, Xiaofan Li, Qi Xiong, Renjing Xu, Zhongyu Li, Wenhao Chai, Enze Xie, Ziwei Wang, Yao Mu, Hao Dong, Wojciech Matusik, Mingyu Ding, Wenbo Ding, Ping Luo, and Masayoshi Tomizuka. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies, 2026b. URL https://arxiv.org/abs/2607.04434.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

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

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

XPolicyLab Community, Tianxing Chen, Yue Chen, Tian Nian, Zijian Cai, Guangyu Chen, Wenwei Lin, Qiwei Liang, Peicheng Xiang, Kailun Su, et al. XPolicyLab: A unified standard and open ecosystem for robot policy evaluation and deployment. *arXiv preprint arXiv:2608.09892*, 2026.

Dexmal. DM0.5. Technical blog, 2026. URL `https://www.dexmal.com/blog/dm0.5`.

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

Ning Gao, Jinliang Zheng, Xing Gao, Haoxiang Ma, Hanqing Wang, Yukai Wang, Jiantong Chen, Zanxin Chen, Shujie Zhang, Mingda Jia, Xuekun Jiang, Zihou Zhu, Xinyu Li, Shuai Wang, Hao Li, Wenzhe Cai, Yuqiang Yang, Xudong Xu, Zhaoyang Lyu, Yao Mu, Tai Wang, Jiangmiao Pang, Jia Zeng, Weinan Zhang, and Chunhua Shen. Ebench: Elemental diagnosis of generalist mobile manipulation policies. *arXiv preprint arXiv:2606.18239*, 2026.

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 arXiv:2604.26694*, 2026.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Junlin Han, Shengbang Tong, David Fan, Minghao Chen, Philip Torr, Filippos Kokkinos, and Mike Lewis. Towards physics of multimodal pretraining: Knowledge flow, modality synergy, early unification, and recipes. *arXiv preprint arXiv:2608.05000*, 2026.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Ryan Hoque, Peide Huang, David J. Yoon, Mouli Sivapurapu, and Jian Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

Yingdong Hu, Haodong Zhu, Boyuan Zheng, Yihang Hu, Tong Zhang, Zunhao Chen, Junming Zhao, Ruiqian Nai, and Yang Gao. Openhlm: An empirical recipe for whole-body humanoid loco-manipulation. *arXiv preprint arXiv:2606.22174*, 2026.

Siqiao Huang, Jialong Wu, Qixing Zhou, Shangchen Miao, and Mingsheng Long. Vid2world: Crafting video diffusion models to interactive world models. *arXiv preprint arXiv:2505.14357*, 2025.

Siqiao Huang, Partha Kaushik, Michael Chen, Hengkai Pan, Kaiwen Geng, Omar Chehab, Fernando Moreno-Pino, and Max Simchowitz. Nano world models: A minimalist implementation of future video prediction. *arXiv preprint arXiv:2605.23993*, 2026.

Saurav Jha, Artem Zholus, Sarath Chandar, et al. Reconstruction or semantics? what makes a latent space useful for robotic world models. *arXiv preprint arXiv:2605.06388*, 2026.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and g0 dual-system vla model. *arXiv preprint arXiv:2509.00576*, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026a.

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

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

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

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why does action chunking improve behavioral cloning performance in robotic control? *arXiv preprint arXiv:2608.02547*, 2026.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *International Conference on Learning Representations*, 2026a.

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

Fanqi Lin, Kushal Arora, Jean Mercat, Haruki Nishimura, Paarth Shah, Chen Xu, Mengchao Zhang, Mark Zolotas, Maya Angeles, Owen Pfannenstiehl, et al. A systematic study of data modalities and strategies for co-training large behavior models for robot manipulation. *arXiv preprint arXiv:2602.01067*, 2026a.

Xiaopeng Lin, Shijie Lian, Bin Yu, Ruoqi Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Yurun Jin, Yukun Shi, Jiyan He, Cong Huang, Bojun Cheng, and Kai Chen. Physbrain: Human egocentric data as a bridge from vision language models to physical intelligence, 2026b. URL https://arxiv.org/abs/2512.16793.

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

Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-horizon manipulation via trace-conditioned vla planning. *arXiv preprint arXiv:2604.21924*, 2026a.

Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, et al. G0.5: One autoregressive stream for robot reasoning and action. *arXiv preprint arXiv:2608.11739*, 2026b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022. URL https://arxiv.org/abs/2201.03545.

Hao Luo, Ye Wang, Wanpeng Zhang, Sipeng Zheng, Ziheng Xi, Chaoyi Xu, Haiweng Xu, Haoqi Yuan, Chi Zhang, Yiqing Wang, et al. Being-h0.5: Scaling human-centric robot learning for cross-embodiment generalization. *arXiv preprint arXiv:2601.12993*, 2026a.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-h0.7: A latent world-action model from egocentric videos. *arXiv preprint arXiv:2605.00078*, 2026b.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

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*, 2026.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Anton Belyi, et al. Mm1: methods, analysis and insights from multimodal llm pre-training. In *European Conference on Computer Vision*, pp. 304–323. Springer, 2024.

Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications. *arXiv preprint arXiv:2503.07137*, 2025.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mahmoud Assran, Koustuv Sinha, Michael Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 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 arXiv:2512.15692*, 2025.

Chaoyi Pan, Giridharan Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. In *International Conference on Learning Representations*, volume 2026, pp. 90575–90614, 2026.

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

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. *arXiv preprint arXiv:2503.09722*, 2025.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv preprint arXiv:2605.18324*, 2026.

Nan Sun, Yuan Zhang, Yongkun Yang, Wentao Zhao, Peiyan Li, Jun Guo, Wenxuan Song, Pengxiang Ding, Runze Suo, Yifei Su, et al. Revisiting embodied chain-of-thought for generalizable robot manipulation. *arXiv preprint arXiv:2606.03784*, 2026.

GigaBrain Team, Angen Ye, Axiang Sun, Can Jin, Chenxi Cheng, Chong Shi, Dengke Shang, Dingqian Zhang, Guan Huang, Guangqiang Wang, et al. Gigabrain-0.7: Scaling embodied foundation models to emergent capabilities with a three-system architecture. *arXiv preprint arXiv:2608.15875*, 2026a.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, et al. Xiaomi-robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026b.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pretraining generalist policy. *arXiv preprint arXiv:2511.16651*, 2025.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.

Shengbang Tong, David Fan, John Nguyen, Ellis Brown, Gaoyue Zhou, Shengyi Qian, Boyang Zheng, Théophane Vallaeys, Junlin Han, Rob Fergus, et al. Beyond language modeling: An exploration of multimodal pretraining. *arXiv preprint arXiv:2603.03276*, 2026.

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

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 arXiv:2605.30280*, 2026a.

Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. *arXiv preprint arXiv:2603.08546*, 2026b.

Zixuan Wang, Yuxin Chen, Yuqi Liu, Jinhui Ye, Pengguang Chen, Changsheng Lu, Shu Liu, Bei Yu, and Jiaya Jia. Vp-vla: Visual prompting as an interface for vision-language-action models. *arXiv preprint arXiv:2603.22003*, 2026c.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. In *International Conference on Learning Representations*, volume 2026, pp. 144731–144838, 2026.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, et al. Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. In *Conference on Robot Learning*, pp. 437–459. PMLR, 2025.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, Qiao Sun, Pengwei Wang, Lingqiao Liu, Yan Wang, Yuxiang Gao, Feras Dayoub, and Haoang Li. 4d-wam: Infusing spatiotemporal awareness into world action models through trajectory fields, 2026a. URL https://arxiv.org/abs/2608.08023.

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 arXiv:2602.11236*, 2026b.

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 arXiv:2603.17240*, 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 arXiv:2604.11757*, 2026b.

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

Yifan Ye, Yankai Fu, Yaoxu Lv, Bohan Hou, Jun Cen, Lingdong Kong, Duo Zheng, Tianxing Chen, Jiaming Liu, Ziang Cao, et al. Data pyramid for embodied manipulation. *arXiv preprint arXiv:2607.24744*, 2026d.

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 arXiv:2606.17846*, 2026a.

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*, 2026b.

He Zhang, Lingzhu Xiang, Haitao Lin, Zeyu Huang, Minghui Wang, Dingyan Zhong, Yubo Dong, Yihao Wu, Yongming Rao, Dongsheng Zhang, et al. Hy-embodied-0.5-vla: From vision-language-action models to a real-world robot learning stack. *arXiv preprint arXiv:2606.14409*, 2026a.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, et al. Native video-action pretraining for generalizable robot control. *arXiv preprint arXiv:2607.08639*, 2026b.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. *arXiv preprint arXiv:2412.18194*, 2024.

Shilong Zhang, He Zhang, Zhifei Zhang, Chongjian Ge, Shuchen Xue, Shaoteng Liu, Mengwei Ren, Soo Ye Kim, Yuqian Zhou, Qing Liu, et al. Both semantics and reconstruction matter: Making representation encoders ready for text-to-image generation and editing. *arXiv preprint arXiv:2512.17909*, 2025a.

Thomas T Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and exploratory data collection yield exponential improvements in behavior cloning for continuous control. *arXiv preprint arXiv:2507.09061*, 2025b.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. Imagewam: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026c.

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

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, volume 2026, pp. 35791–35820, 2026a.

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

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8152–8162, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

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

APPENDIX

This appendix provides supplementary analyses and implementation details supporting the main paper:

- Appendix A discusses limitations of our work and directions for future research.
- Appendix B situates OpenWAM within prior work on world–action models, open research ecosystems, and the empirical study of model design.
- Appendix C specifies the four components of OpenWAM-Infra in full: the composable model, the training runtime, the deployment runtime, and the evaluation protocol.
- Appendix D gives the complete experimental descriptions and extended analyses behind the controlled studies of OpenWAM-Study.
- Appendix E provides the complete specification of OpenWAM-$\boldsymbol{\alpha}$, its pretraining data mixture and curation pipeline, and extended analyses of its results.
- Appendix F documents the pretraining configuration and the dataset-specific SFT configurations used during post-training.
- Appendix G details the task setups and evaluation protocols of the real-world experiments.
- Appendix H reports the full per-benchmark simulation scores behind Figure 9.

A LIMITATIONS AND FUTURE WORK

While OpenWAM provides a fully-open, systematic exploration towards world–action model pretraining, it has several limitations and opens up interesting future directions worth exploring.

1. **Training phases.** We mostly focus on the embodied pretraining phase of world–action modeling. Post-training and adaptation methods can lead to significant improvements for embodied foundation models, and the empirical recipe as well as underlying mechanisms for these methods remain open questions.
2. **Architecture.** Across the six architecture variants currently supported by OpenWAM, we mostly explore modality fusion through cross-modality attention or hard-routed MoE. Drawing experience from the Unified Multimodal Model (UMM) community, we encourage future work to explore more native modality-fusion techniques, such as tokenization-phase early fusion and soft-routed MoE.
3. **Pretraining data mixture.** We did not include UMI-style (e.g., UMI (Chi et al., 2024), DexUMI (Xu et al., 2025)) collected data. In theory, UMI-style data offers task and scene diversity comparable to human egocentric videos, which is a crucial component for out-of-domain generalization capabilities. Co-training with data that contain robot-executable actions but are diverse in scene and task level, which can either be collected through UMI-style interfaces or post-processing pipelines, may offer a more data-efficient path towards autonomous embodied machine intelligence. In addition, we look forward to further breakthroughs in simulation for embodied AI: simulation can natively generate robot manipulation data with diverse scenes and realistic motion trajectories, unconstrained by the time and labor costs of the physical world, and thus holds unbounded potential for scaling robot data by orders of magnitude.
4. **Visual encoder.** Weighing the compression of candidate encoders in both the temporal and the token dimension, we ultimately adopt Wan2.2-VAE as the final encoder of OpenWAM — a choice that reflects the best trade-off currently available rather than an optimal solution: our evaluations reveal that pixel-reconstruction encoders such as Wan2.2-VAE are not sufficiently robust to viewpoint, noise, and scene variations. A latent representation that is compact while carrying sufficient environment information is still needed to push WAM performance further, and merits deeper exploration.

B RELATED WORK

World–Action Models. World models (LeCun et al., 2022; Ha & Schmidhuber, 2018) learn predictive structure from observations and have long supported control through planning (Zhou et al., 2024; Maes et al., 2026; Huang et al., 2026), model-based reinforcement learning (Hafner et al., 2019; M. Moerland et al., 2023), and policy evaluation (Huang et al., 2025; Wang et al., 2026b).

World–Action Models (WAMs) (Ye et al., 2026c; Pai et al., 2025; Li et al., 2026b) more directly connect this predictive capacity to executable behavior by serving as a policy model. Whereas VLA models (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024) primarily inherit semantic and linguistic knowledge from vision–language pretraining (Beyer et al., 2024; Bai et al., 2025), WAMs initialize from video-generative priors so that action learning begins with a model with rich visual dynamical priors (Wan et al., 2025; Ali et al., 2025). Yet existing systems remain largely *monolithic*, where changes in model architecture, training procedure, data recipe, and sampling schedule are often coupled. Consequently, it remains unclear which components transfer world knowledge, which interactions create world–action synergy, and which benefits persist across domains. **OpenWAM** exposes these coupled choices as controlled variables and organizes them around precisely these three questions.

Open Research Ecosystems for Generalist Robot Policy Learning. Open models and codebases have made generalist robot learning increasingly accessible. OpenVLA (Kim et al., 2024) established an open-weight pretrained baseline, while StarVLA (Community, 2026) provides a modular and performant platform for varied design choices. StarVLA-α (Ye et al., 2026b) complements this breadth with a pretrained model of minimalist design, and XPolicyLab (Community et al., 2026) contributes a unified standard and open ecosystem for policy evaluation and deployment. These efforts have significantly reduced development complexity in the VLA research community; however, in the WAM community, such open research ecosystems remain largely absent. A modular system in this realm accompanied by a strong pretrained model would help democratize research, as well as serve as a principled foundation for understanding and scaling world–action model pretraining.

Towards a Scientific Understanding of Model Design. A growing line of work treats model design as an empirical science (Allen-Zhu, 2026; Karras et al., 2022; McKinzie et al., 2024; Liu et al., 2022; Wen et al., 2026): decomposing a complex system into controlled variables, testing the mechanisms behind observed gains, deriving a recipe, and validating whether it survives scale. In multimodal learning, Cambrian-1 (Tong et al., 2024) and Beyond Language Modeling (Tong et al., 2026) systematically study visual representations, modality-specific capacity, data composition, and unified pretraining; Towards Physics of Multimodal Pretraining (Han et al., 2026) further isolates knowledge flow, synergy versus competition, and the timing of modality unification. In robot learning, analyses around Action Chunking (Simchowitz et al., 2025; Zhang et al., 2025b; Lazzati et al., 2026) and Generative Control Policies (Pan et al., 2026) have substantially reshaped the community's understanding of these topics. At the data and system level, Large Behavior Models (Barreiros et al., 2026), LBM co-train (Lin et al., 2026a), StarVLA-α (Ye et al., 2026b), and OpenHLM (Hu et al., 2026) similarly use controlled comparisons to study multitask transfer, heterogeneous supervision, action design, and embodiment interfaces. **OpenWAM** brings this methodology to WAMs: **OpenWAM-Infra** builds the substrate for controlled experiments, **OpenWAM-Study** turns them into controlled scientific questions about inheritance, synergy, and scaling, and **OpenWAM-α** scales the resulting recipe under heterogeneous multi-domain pretraining.

C OpenWAM-Infra: Implementation Details

This section expands the summary of Section 2 into the full specification of OpenWAM-Infra. We follow the same order as the main text and treat the four components in turn: the composable model that assembles a concrete architecture from interchangeable modules (Appendix C.1), the training runtime that trains every such architecture under one trainer (Appendix C.2), the deployment runtime that serves every resulting checkpoint through one policy server (Appendix C.3), and the evaluation protocol through which every benchmark reaches that server (Appendix C.4). For each component we give the code contract that keeps it decoupled from the others, enumerate the module variants and options OpenWAM-Infra currently ships, and state the formulations that the main text reports only in summary.

C.1 Composable Model

As shown in Figure 11, OpenWAM-Infra organizes a World–Action Model (WAM) as three classes of interchangeable modules that a composition rule C assembles into a concrete **architecture**, written $C(\mathcal{E}, \mathcal{S}, \mathcal{M})$:

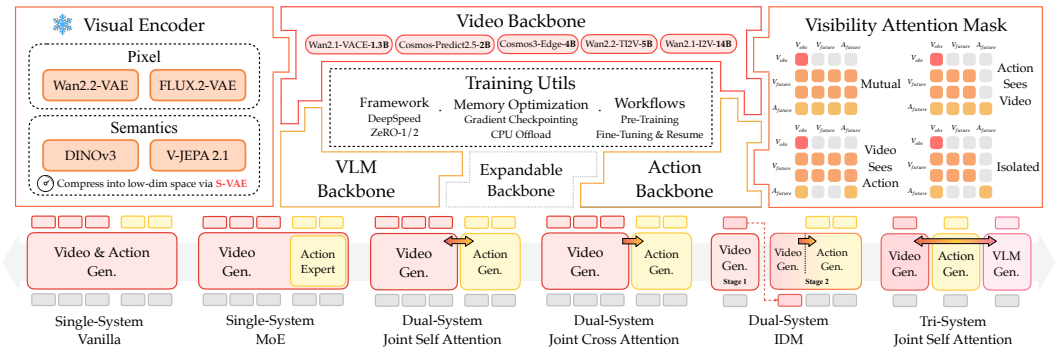

Figure 11: **OpenWAM Model Infra.** *Top*: the three classes of interchangeable modules: the visual encoder $\mathcal{E}$ (left); the stream backbones $\mathcal{S}$ (middle); and the visibility attention mask $\mathcal{M}$ (right). The central Training Utils panel summarizes the utilities of the training runtime (Appendix C.2). *Bottom*: the composition rule C assembles the modules into six architecture variants across the Single-System, Dual-System, and Tri-System families.

- **Visual Encoder** $\mathcal{E}$: It maps observations into the latent sequences the world stream predicts;
- **Stream Backbones** $\mathcal{S}$: It processes the model's token streams: a world stream $\mathcal{W}$, an action stream $\mathcal{A}$, optionally an understanding stream $\mathcal{U}$, and any further streams a design may introduce; distinct streams may share a single backbone;
- **Visibility Attention Mask** $\mathcal{M}$: It specifies the attention relations among streams and tokens, i.e. which tokens may attend to which, both within and across streams.

Visual Encoder. The encoder $\mathcal{E}$ maps observations into the latent sequence the world stream predicts and is always kept frozen; different encoders capture different information in their latents and in turn induce different world-stream behavior. In most cases, a video backbone is accompanied by a natively matched encoder, in which case the pretrained parameters of the base DiT are reused directly, retaining the full benefit of pretraining. Beyond this default, OpenWAM-Infra additionally supports pluggable encoders: swapping $\mathcal{E}$ alters the latent representation to be predicted, and the base DiT can further be re-initialized to exclude the influence of pretrained parameters. Currently, OpenWAM-Infra provides two reconstructive encoders, trained on pixel-reconstruction objectives: Wan2.2-VAE (Wan et al., 2025) and FLUX.2-VAE (Black Forest Labs, 2025), and two representation encoders (Zheng et al., 2026a): DINOv3 (Siméoni et al., 2025) and V-JEPA 2.1 (Mur-Labadia et al., 2026), along with an optional S-VAE (Zhang et al., 2025a) module that compresses the latent dimension. These capabilities together support the study of visual representations in Section 3.1.

Stream Backbones. The backbones in $\mathcal{S}$ are laid out in and around the central panel of Figure 11. The *video backbone* ($\mathcal{W}$) predicts the temporal evolution of future world visual states, while the *action backbone* ($\mathcal{A}$) predicts the upcoming action chunk (Zhao et al., 2023); together they form the world–action core of the model. The optional *VLM backbone* ($\mathcal{U}$) supplements this core with semantic understanding of the current observation. Beyond these, the backbone roster is itself **expandable**: further backbones can be registered to execute any additional streams a design may introduce. Regardless of type, every backbone executes through one code contract that decomposes its forward pass into three stages:

- `prepare`, invoked once before the stack, which embeds the inputs into the initial token state;
- `per-layer block step`, invoked once per layer, which advances this state through one transformer layer;
- `finalize`, invoked once after the stack, which maps the final state to the stream's prediction.

Each layer may further split its block step into a `pre-attention` half, which emits the layer's queries, keys, and values, and a `post-attention` half, which consumes the attention output, so that the attention between the two halves can be computed jointly across streams.

For the video backbone, OpenWAM-Infra supports five pretrained video generation models with increasing model parameters, namely Wan2.1-VACE-1.3B, Cosmos-Predict2.5-2B, Cosmos3-Edge-4B, Wan2.2-TI2V-5B, and Wan2.1-I2V-14B (Wan et al., 2025; Ali et al., 2025). For VLM backbones, OpenWAM-Infra currently supports only the Qwen3-VL family (Bai et al., 2025). For the action backbone, OpenWAM-Infra offers two options: a separate set of parameters residing in ActionDiT, or a shared video backbone in which action tokens join the video token sequence and are processed jointly.

Visibility Attention Mask. The mask $\mathcal{M}$ governs the information flow with the mixed self-attention through which streams interact: it factorizes into intra-modality and cross-modality blocks, granting attention where tokens reinforce one another and withholding it where their mutual influence must be isolated. Over the video and action modalities, this factorization reads

$$\mathcal{M} = \begin{pmatrix} \mathcal{M}_{V\leftarrow V} & \mathcal{M}_{V\leftarrow A} \\ \mathcal{M}_{A\leftarrow V} & \mathcal{M}_{A\leftarrow A} \end{pmatrix}, \tag{1}$$

where the block $\mathcal{M}_{X\leftarrow Y}$ specifies whether tokens of modality X may attend to tokens of modality Y. OpenWAM-Infra fixes the two intra-modality blocks: $\mathcal{M}_{V\leftarrow V}$ adopts *first-frame causal* attention, in which noisy frames attend to one another and to the clean first frame while the clean frame attends only to itself, shielding clean conditioning from noise; $\mathcal{M}_{A\leftarrow A}$ adopts *bidirectional* attention, in which the noisy action tokens of a chunk are mutually visible so that the predicted actions inform one another. The two cross-modality blocks then define the four attention mask modes that OpenWAM-Infra supports, as drawn in the right panel of Figure 11: *mutual* enables both $\mathcal{M}_{A\leftarrow V}$ and $\mathcal{M}_{V\leftarrow A}$, so the two modalities attend to each other; *action-sees-video* enables only $\mathcal{M}_{A\leftarrow V}$, letting actions read the predicted world while leaving video generation undisturbed; *video-sees-action* enables only $\mathcal{M}_{V\leftarrow A}$, the reverse; and *isolated* disables both, denoising the two modalities independently. Building on this native support, Section 3.2 later compares these modes under controlled settings.

Architectures. The composition rule C specifies where and how information crosses streams, and it does so purely through the execution contract above: it sequences the **prepare**, **per-layer block**, and **finalize** stages of the participating backbones and, when interaction must occur inside attention, splits the block step into its **pre-attention** and **post-attention** halves to substitute the attention computation itself, never modifying backbone internals. Composition rules therefore carry no parameters of their own; all learned capacity resides in the stream backbones. A concrete architecture is a choice $C(\mathcal{E}, \mathcal{S}, \mathcal{M})$; the designs currently supported fall into three families, laid out left to right in the bottom row of Figure 11.

- **Single-System.** This family comprises only the video backbone: the action backbone takes the shared form and injects its tokens into the video sequence, so one transformer processes both modalities with most parameters shared; representative systems include Cosmos Policy (Kim et al., 2026b) and DreamZero (Ye et al., 2026c). OpenWAM-Infra provides two variants, differing in modality-specific capacity:
  - *Vanilla* processes video, action, and proprioceptive tokens as one sequence through the same attention and dense feed-forward blocks, providing no modality-specific capacity.
  - *MoE* retains the shared sequence and self-attention but hard-routes action tokens to a dedicated feed-forward expert while video tokens follow the default path, adding modality-specific capacity without separating the streams (Mu & Lin, 2025).
- **Dual-System.** This family comprises an independent video backbone and action backbone, the latter a dedicated ActionDiT: the two streams hold separate sets of parameters and are connected through self- or cross-attention; representative systems include Fast-WAM (Yuan et al., 2026b) and LingBot-VA (Li et al., 2026b). OpenWAM-Infra provides three variants, differing in how the two streams communicate:
  - *Joint self-attention* merges the hidden states of the two streams into a joint sequence at designated bridge layers, allowing bidirectional token-level interaction before the states return to their streams.
  - *Joint cross-attention* lets the action stream query video features through video-to-action cross-attention at the bridge layers, trained end-to-end so that action-learning gradients up-

date the video stream; optionally, gradients are detached at the video features to isolate action learning from video parameter updates.
  - *IDM* formulates the action module as an inverse-dynamics model conditioned on video features, trained with teacher-forced video states and run in two inference stages: the video trajectory is generated first and the actions are predicted from it.
- **Tri-System.** This family extends the dual layout with a VLM backbone, in which a frozen vision–language model feeds a separate trainable understanding stream; representative systems include Motus (Bi et al., 2026). OpenWAM-Infra provides a single variant:
  - *Joint self-attention* extends the joint sequence to all three streams, which exchange information while retaining stream-specific parameters; the understanding stream joins as a read-only tail that the other streams may attend to while it attends only to itself.

Within each architecture $C(\mathcal{E}, \mathcal{S}, \mathcal{M})$, every module (the visual encoder, the stream backbones, and the visibility attention mask) is instantiated from a registry, orthogonally to the composition rule: every combination is assembled from configuration, and neither the trainer nor the policy server is aware of which architecture is running.

C.2 Training Runtime

Training Formulation. OpenWAM-Infra trains every architecture under one trainer, against one sample contract and one joint flow-matching objective. The trainer never inspects architecture internals: it asks the selected architecture to prepare its own inputs and run its own forward pass, then optimizes the objective on the resulting predictions. Define a sample as $(\ell,\ \mathbf{o}_{1:T},\ \mathbf{a}_{1:H},\ \mathbf{q},\ \mathbf{m})$, a language instruction, a video window, an action chunk, an optional proprioceptive state, and a per-dimension validity mask. During input preparation, the architecture's visual encoder $\mathcal{E}$ encodes $\mathbf{o}_{1:T}$ into latents $\mathbf{z}$, and ℓ, optionally joined by $\mathbf{q}$, becomes the context $\mathbf{c}$. Throughout the paper, $t = 0$ denotes pure noise and $t = 1$ clean data. Each stream is noised to its own timestep, t_v for video and t_a for actions, yielding the interpolants $\mathbf{z}^{t_v} = t_v\,\mathbf{z} + (1 - t_v)\,\boldsymbol{\epsilon}_v$ and $\mathbf{a}^{t_a} = t_a\,\mathbf{a} + (1 - t_a)\,\boldsymbol{\epsilon}_a$ with Gaussian noise $\boldsymbol{\epsilon}_v, \boldsymbol{\epsilon}_a$; one joint forward pass of the architecture $(\hat{\mathbf{v}}_z, \hat{\mathbf{v}}_a) = \mathbf{v}_\theta(\mathbf{z}^{t_v}, \mathbf{a}^{t_a}, t_v, t_a, \mathbf{c})$ predicts both velocities, and the objective takes the form

$$\mathcal{L} = \lambda_v\,\mathbb{E}_{t_v,\boldsymbol{\epsilon}_v}\Big[w(t_v)\,\big\|\hat{\mathbf{v}}_z - (\mathbf{z} - \boldsymbol{\epsilon}_v)\big\|_2^2\Big] + \lambda_a\,\mathbb{E}_{t_a,\boldsymbol{\epsilon}_a}\Big[w(t_a)\,\big\|\mathbf{m} \odot (\hat{\mathbf{v}}_a - (\mathbf{a} - \boldsymbol{\epsilon}_a))\big\|_2^2\Big], \qquad (2)$$

where λ_v, λ_a and $w(\cdot)$ weight the streams and the timesteps, while the validity mask $\mathbf{m}$ restricts the action term to the coordinates an embodiment actually populates, and clean conditioning frames are excluded from the video term. Because t_v and t_a are sampled *independently*, training covers the entire (t_v, t_a) noise plane; any inference schedule, whether it denoises the two streams synchronously at a shared timestep or asynchronously with one stream leading the other, traces a path through this plane and thus remains in-distribution.

Training Utilities. As shown in the Training Utils panel at the center of Figure 11, three core utilities support OpenWAM-Infra training:

1. **Framework.** OpenWAM-Infra integrates DeepSpeed ZeRO (stage 1 or 2) through Accelerate and supports mixed precision (bf16 by default), gradient accumulation, and gradient clipping; a single entry point scales from single-GPU runs to multi-node jobs.
2. **Memory optimization.** To reduce memory consumption, OpenWAM-Infra provides gradient checkpointing, with optional CPU offload of the checkpointed activations, and optimizer-state offload to CPU.
3. **Workflows.** OpenWAM-Infra supports three training workflows. *Pretraining* starts a fresh run. *Fine-tuning* starts a new run initialized from a previous checkpoint: the architecture is rebuilt from the checkpoint's own record, the new configuration is layered on top, and the identity of the modules the weights belong to is protected from override. *Resume* continues the same run exactly: the full optimizer and scheduler state is restored, training re-enters the data stream at the recorded position, and the run refuses to continue if the dataset's normalization statistics diverge from those recorded with the run.

Self-Contained Checkpoints. A self-contained checkpoint comprises three parts: the model weights, the fully resolved configuration with every module's reconstruction specification (and artifacts such as tokenizers) merged in, and the action-normalization statistics. Fine-tuning, resume, and deployment all rebuild the architecture from this record before loading parameters; at deployment, a missing normalization record is a hard error. An evaluation therefore cannot silently change the encoder, the action layout, or the scaling of a trained model, and the checkpoint is exactly what the deployment runtime serves.

C.3 Deployment Runtime

Every checkpoint is served by one policy server, which rebuilds the architecture from its self-contained record and keeps two choices orthogonal: when inference runs (the inference mode) and how the two streams are denoised (the denoising schedule). Figure 12 illustrates the two choices in panels (a) and (b), respectively.

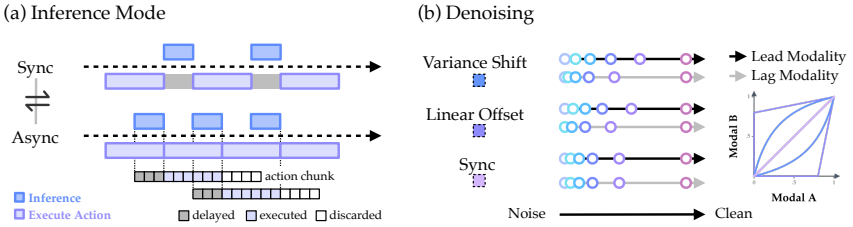

Figure 12: **Inference modes and denoising schedules of the deployment runtime.** (a) Illustration of the synchronous and asynchronous inference modes. (b) Illustration of the three denoising schedules (variance shift, linear offset, and sync); five denoising steps are drawn for illustration, and circles of the same color denote the timesteps that the two modalities reach at the same denoising step.

Inference Modes. OpenWAM-Infra provides two inference modes over a common buffer mechanism (Figure 12(a)): each inference produces an action chunk that is buffered, and the server pops one action per request. Under *synchronous* inference, the server blocks on a fresh inference whenever the buffer empties, so execution stalls for the inference latency. Under *asynchronous* inference, let H denote the length of the predicted chunk, $n \le H$ the inference horizon, and $d < n$ the lead threshold in steps, defaulting to $n/2$. Once only d buffered actions remain, a single background worker prefetches the next chunk while those actions keep executing. The adopted chunk then splits, in order, into a *delayed* prefix of d actions, already covered by the previous buffer while inference ran and therefore skipped; an *executed* window of the next n actions, which becomes the new buffer; and a *discarded* tail of the remaining $\max\{H - d - n, 0\}$ actions. The two modes are indistinguishable to the client: every request is one observation in, one action out.

Denoising Schedules. A denoising schedule is a path $\tau = \{(t_v^i, t_a^i)\}_{i=0}^{N}$ through the joint noise plane, with $t = 0$ pure noise and $t = 1$ clean data as in Appendix C.2; Figure 12(b) draws the three schedules that OpenWAM-Infra supports. Under the *sync* schedule, both streams advance in lockstep along the diagonal: each step performs one joint forward pass and a coupled Euler update in which each stream moves by its own timestep increment; the video latents $\mathbf{z}$ follow $\mathbf{z}^{t_v^{i+1}} = \mathbf{z}^{t_v^i} + (t_v^{i+1} - t_v^i)\,\hat{\mathbf{v}}_z$, and the action chunk $\mathbf{a}$ follows $\mathbf{a}^{t_a^{i+1}} = \mathbf{a}^{t_a^i} + (t_a^{i+1} - t_a^i)\,\hat{\mathbf{v}}_a$. Asynchronous schedules let one stream lead through two composable families (Baade et al., 2026),

$$f_\alpha(s) = \frac{\alpha s}{1 + (\alpha - 1)s}, \qquad h_o(s) = \max\left\{\frac{s - o}{1 - o}, 0\right\}, \tag{3}$$

where $s = i/N$ denotes global progress, the *variance shift* curve f_α lifts the leading stream above the diagonal for $\alpha > 1$ so that it reaches clean data earlier, and the *linear offset* h_o holds the lagging stream at pure noise until global progress exceeds o. Assigning the lead to the world stream or to the action stream yields the *video-lead* and *action-lead* regimes, and $(\alpha, o) = (1, 0)$ recovers the synchronized diagonal exactly: synchronous serving is a special case rather than a separate

code path, and every asynchronous run has an aligned baseline. Because training samples the two timesteps independently (Appendix C.2), every such path stays in-distribution. Independently of the schedule shape, each stream's timestep warp is a backbone property stored in the checkpoint and reused at inference, so the training and serving noise grids cannot drift.

Acceleration. OpenWAM-Infra provides four serving-side accelerations, each independently configurable.

• **Prompt-embedding cache**: a server-lifetime cache maps each prompt to its text-encoder embeddings, removing the text encoder from the per-request path.

• **Video-decode skip**: control consumes actions rather than pixels, so the serving path can skip VAE video decoding entirely.

• **Compilation**: each architecture registers a fixed-shape `torch.compile` path for its inner joint denoising loop, replayed under CUDA graphs to eliminate per-layer launch overhead; the first request carries the compilation warmup.

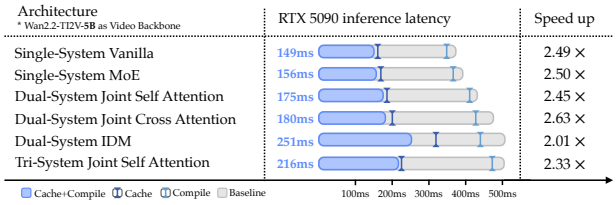

Figure 13: **Serving latency across architectures.** Inference latency with Wan2.2-TI2V-5B as the video backbone on an RTX 5090. The prompt-embedding cache and video-decode skip are enabled by default; the figure ablates compilation and the DiT velocity cache.

• **DiT velocity cache**: when the recent velocity predictions of *both* streams are similarity-stable (cosine similarity above a threshold), the next joint forward pass is skipped and the cached velocities are integrated instead, with a bounded number of consecutive skips, following the cross-step reuse of Ye et al. (2026c).

With Wan2.2-TI2V-5B as the video backbone, Figure 13 illustrates the resulting inference speedups across the different architectures.

C.4 Evaluation Protocol

OpenWAM-Infra evaluates trained checkpoints through the policy server of Appendix C.3: each benchmark connects as a client, sends observations, and executes the actions returned by the server, as shown in Figure 14.

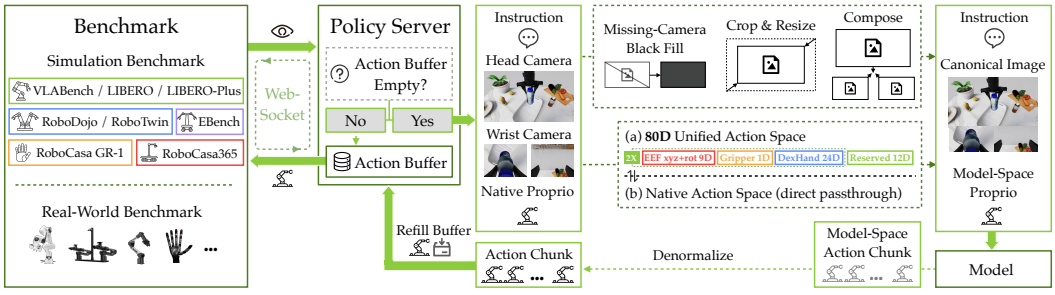

Figure 14: **Evaluation protocol of OpenWAM-Infra.** Benchmarks connect to the policy server as thin clients over WebSocket. The server canonicalizes each observation, maps the proprioceptive state into the model-side action representation (the 80-D unified action space or the benchmark's native action space), and denormalizes the predicted action chunk back to native physical units before returning actions.

Client–Server Pipeline. Benchmarks reach the deployment runtime as thin clients over one persistent WebSocket connection and import nothing from the model or training stack. At control step k, the client sends an observation $(\mathbf{o}_k, \ell, \mathbf{q}_k)$: up to three camera views $\mathbf{o}_k$, with the head view required and the wrist views optional; the language instruction ℓ; and optionally the raw robot state $\mathbf{q}_k$. The response is a single action $\mathbf{a}_k$ in the robot's native physical units. Every model-facing conversion runs server-side, driven by the self-contained checkpoint of Appendix C.2: as laid out

in Figure 14, the views are cropped, resized, and composed into the canonical image layout the checkpoint was trained on, with missing cameras filled by black frames, and $\mathbf{q}_k$ is normalized and mapped into the model-side action representation defined below. Inference under the serving stack of Appendix C.3 then yields a model-space action chunk $\hat{\mathbf{a}}_{1:H}$, which is mapped back and denormalized into native units before it refills the action buffer from which the server answers requests; normalized values therefore never reach a robot. Between episodes, a single reset request clears all per-episode executor state.

Benchmark Suite. OpenWAM-Infra currently integrates eight simulation benchmarks: LIBERO and LIBERO-Plus (Liu et al., 2023; Fei et al., 2025), VLABench (Zhang et al., 2024), RoboTwin2.0 (Chen et al., 2025), RoboDojo (Chen et al., 2026b), RoboCasa365 (Nasiriany et al., 2026), RoboCasa-GR1 (NVIDIA et al., 2025; Nasiriany et al., 2024), and EBench (Gao et al., 2026), together spanning single-arm and bimanual tabletop manipulation, dexterous-hand humanoid control, and mobile manipulation. Because the protocol exchanges only images, text, and action vectors, real-robot platforms connect through exactly the same interface as the simulators. Each bundled adapter reproduces the observation preprocessing of its benchmark's training reader, so evaluation-time views match the training distribution; integrating a new benchmark amounts to writing such an adapter, leaving the model and both runtimes untouched.

Action Space Definition. Actions cross this pipeline in one of two representations (Figure 14). By default, every benchmark keeps its *native* action space — RoboTwin2.0, for instance, is served in either a 14-D joint space or a 20-D bimanual end-effector space, and LIBERO in a 10-D end-effector space — so a checkpoint trained on a single benchmark passes actions straight through. Training one model across embodiments, however, requires a single action head over bodies whose native layouts differ in both width and semantics. OpenWAM-Infra therefore also defines a *unified action space* $\mathbf{u} \in \mathbb{R}^{80}$ with fixed slot semantics: two mirrored 34-D arm blocks, each comprising the end-effector position (3), a 6D rotation (6), the gripper (1), and a dexterous hand (24), followed by 12 reserved slots for embodiment-specific channels such as the mobile bases of EBench and RoboCasa365. Since the slot semantics are fixed, the structure that embodiments share lands on the same coordinates. Each dataset declares an index map π from its native dimensions into these slots, with normalization applied *before* scattering and inverted *after* gathering,

$$\mathbf{u} = \mathrm{Scatter}_{\pi}\big(\mathrm{Norm}(\mathbf{a})\big), \qquad \mathbf{a} = \mathrm{Norm}^{-1}\big(\mathrm{Gather}_{\pi}(\mathbf{u})\big), \tag{4}$$

and incoming proprioception traverses the same map in the forward direction. The validity mask $\mathbf{m}$ of Equation (2) marks exactly the mapped slots, so unmapped coordinates receive no gradient during training and remain on their analytic noise path at inference.

D Extended Results of OpenWAM-Study

This section provides the complete experimental descriptions behind Section 3: the evaluation protocol, the setup and full analysis of each controlled study, and the cross-domain pretraining protocol. The figures and tables referenced here are those of the main text.

D.1 Evaluation Protocol

Evaluation Protocol. In this section, we use **RoboTwin2.0** (Chen et al., 2025), a widely-adopted bi-manual manipulation benchmark spanning over 50 tasks, as our evaluation environment. In our experiments, we evaluate under two settings: (1) **In-Domain Performance (RoboTwin2.0-Full)**: Following Bi et al. (2026), we train our model with an entire multi-task data corpus of 2,500 demonstrations collected in clean scenes and 25,000 demonstrations collected under heavy scene randomization, and evaluate under clean and randomized environments separately. (2) **Out-of-Domain Generalization (RoboTwin2.0-Clean2Random)**: Following Yuan et al. (2026a), we train our model on clean data only and evaluate under clean and randomized environments separately. Since the training mixture has never seen randomized scene configurations, it serves as a proxy for evaluating the models' generalization capabilities to novel scenes. We use success rate as our metric.

D.2 Generative World Priors

To understand whether and to what extent generative world priors facilitate WAM performance, we compare four video generation backbones with variable parameter counts: Wan2.1-VACE-1.3B (Wan et al., 2025), Cosmos-Predict2.5-2B (Ali et al., 2025), Wan2.2-TI2V-5B (Wan et al., 2025), and Wan2.1-I2V-14B (Wan et al., 2025). We instantiate our World–Action Model with a Dual-System architecture consisting of a video generation module and an action generation module connected with joint self-attention, in which the video generation module parameters are copied from the pre-trained video generation model. Results are evaluated on RoboTwin2.0-Full.

Across the four tested backbones, the average success rate of the resulting WAM improves consistently with video generation backbones of increasing capacity (Figure 3). Wan2.1-I2V-14B performs best, while Wan2.2-TI2V-5B trails it by only 1.40 points, even though the former has nearly **3x** the parameter count. Balancing performance against training and deployment efficiency across the four backbones, we ultimately adopt the 5B model as the default for the remaining controlled studies. Since these backbones also differ in architecture, pretraining data, and objective, this comparison establishes backbone choice as a consequential design variable without attributing the entire gain to parameter count alone. The 5B default also keeps subsequent action-side ablations tractable while holding the inherited world prior fixed.

D.3 Visual Representation Priors

Another important source of world knowledge comes from the latent representation space induced by vision encoders. Broadly speaking, these encoders fall into two categories: (1) **Reconstructive Encoders**: the objective of these encoders is compression, trained solely with pixel-reconstruction; (2) **Representation Encoders**: grounded by self-supervised or multimodal representation learning, these encoders learn semantically structured visual features that provide a basis for visual understanding. Motivated by recent advances in generative and world modeling with representation encoders (Zheng et al., 2026a; Singh et al., 2026; Zhou et al., 2024; Jha et al., 2026; Lyu et al., 2026), we question the design of latent space in the context of world–action models.

In this set of experiments, we choose representative encoders from both categories. For Reconstructive Encoders, we use Wan2.2-VAE (Wan et al., 2025), a state-of-the-art video encoder with a 4x temporal compression rate, and FLUX.2-VAE (Black Forest Labs, 2025), an advanced image encoder yielding highly performant image generation models built on its latent space. For Representation Encoders, we use DINOv3 (Siméoni et al., 2025), the newest generation of DINO (Caron et al., 2021), a classical vision encoder learned through self-supervised learning; and V-JEPA 2.1 (Mur-Labadia et al., 2026), a dense feature encoder based on joint embedding predictive architectures (LeCun et al., 2022). To isolate the performance gain from visual representations alone, we inherit the model architecture of Wan2.2-TI2V-5B, but randomly initialize its model weights. For a fair comparison, we apply the same 4x temporal compression as Wan2.2-VAE to FLUX.2-VAE, DINOv3, and V-JEPA 2.1: since none of these encoders natively performs temporal compression, we impose the 4x downsampling by averaging the features of every four consecutive frames.

Since modern Diffusion Transformer architectures (Peebles & Xie, 2023) are optimized mostly for reconstructive encoders, naive adoption of representation encoders, which produces high-dimensional latents (e.g., 768-D features for DINOv3 and 1024-D for V-JEPA 2.1), can lead to poor performance due to architectural incompatibility (Zheng et al., 2026a). Following Jha et al. (2026), we include a variant for representation encoders, where we train an S-VAE (Zhang et al., 2025a) adapter that converts the high-dimensional features produced by representation encoders to lower-dimensional vectors (48-D in this experiment, matching the latent dimension of Wan2.2-VAE).

As shown in Figure 4, representation encoders can yield performance on par with or stronger than reconstructive encoders for world–action modeling with the help of S-VAEs. Specifically, while naive adoption of representation encoders yields worse performance than models trained with the reconstructive encoder FLUX.2-VAE, with dimension contraction using S-VAE, the representation encoders' contracted variants (DINOv3 w/ SVAE, V-JEPA 2.1 w/ SVAE) significantly outperform FLUX.2-VAE. DINOv3 w/ SVAE achieves nearly on-par performance with Wan2.2-VAE, and we attribute the remaining slim margin to two native advantages of Wan2.2-VAE: it is a reconstructive encoder specifically suited to the video backbone architecture, and its temporal compression is learned natively by the encoder rather than imposed through frame averaging.

Taken together, what determines the quality of a WAM latent space is not the categorical divide between reconstructive and representation encoders, but the operational properties of the latents themselves: compactness (in both the temporal and the token dimension) and rich world information. Priors in representation encoders can thus be inherited to build highly performant WAMs with the help of **temporal compression** and **dimension contraction**. We also encourage active research into building representation encoders with native temporal compression, which in turn may lead to even better prior inheritance.

D.4 Architectural Capacity

A central question in world–action modeling is how much action-specific capacity a WAM requires and how strongly its video and action streams should be separated. The three architecture families of Appendix C.1 span precisely this capacity axis, and we evaluate all six of their variants, instantiating joint cross-attention both end-to-end and with gradients detached at the video features, yielding seven baselines (Table 1).

Results. As shown in Table 1, with increasing architecture capacity, performance from single- to dual- and tri-system continuously improves. Joint self-attention is the strongest dual-system variant, while the tri-system model achieves the best overall performance. Balancing performance with architectural complexity, and isolating the interaction between world knowledge and action learning from the potential influence of the VLM's understanding features, we therefore adopt dual-system joint self-attention for the remaining experiments, so that the subsequent findings reflect this interaction alone.

D.5 Training-Time Information Flow

The architectural comparison selects joint self-attention as the interface between the world and action streams, but joint attention alone does not specify which information flow creates the best synergy. We compare four information flow strategies at training time, controlled by attention masking: **Isolated**, with no cross-stream communication; **Video Sees Action**, which exposes action features to the world stream; **Action Sees Video**, which exposes world features to the action stream; and **Mutual**, which enables both directions.

The comparison separates cleanly according to whether the action stream can access world features (Table 2). Isolated and video-sees-action masks underperform by roughly five points, whereas action-sees-video and mutual visibility both retain strong performance. World-to-action flow is therefore necessary in this setting. By contrast, adding the reverse action-to-world path changes the from-scratch result only marginally, leaving action-sees-video and mutual visibility as two viable masks to revisit after pretraining.

D.6 Inference-Time Information Flow

Training-time attention masking implements *full masking*, while the inference-time denoising schedule provides a softer information gate, i.e. *partial masking*. For this ablation, we fix mutual visibility and train the video and action streams with independently sampled noise levels, covering a two-dimensional space of joint noise states.

We instantiate the schedule abstraction of Appendix C.3: synchronized denoising follows the diagonal $(t_v^i, t_a^i) = (s_i, s_i)$, while the variance-shift and linear-offset families of Equation (3) let either stream lead (Baade et al., 2026). We evaluate $\alpha \in \{4, 8, 16, 32\}$ and $o \in \{0.2, 0.4, 0.6, 0.8\}$ in both leading directions.

Results. Synchronized denoising performs best, and no asynchronous schedule improves performance, regardless of which stream leads or how relative progress is parameterized (Figure 5). With variance-shift schedules, video-leading outperforms action-leading schedules, whereas with linear-offset schedules, action-leading outperforms video-leading schedules. This suggests that while explicit video-to-action information flow is necessary at training time, enforcing such priors through the inference-time denoising schedule does not yield gains, supporting the representation learning

hypothesis of world–action modeling (Yuan et al., 2026b) as opposed to an implicit planning-then-IDM schedule at test time (Ye et al., 2026c).

D.7 Cross-Domain Pretraining

Controlled Transfer Protocol. All runs keep the backbones, optimization budget, and inference procedure fixed — the dual-system joint self-attention architecture with synchronized denoising selected above — and vary only whether and how the model is pretrained with embodiment data; the information-flow mask is revisited in the final ablation. Two complementary protocols serve the evaluation: **RoboTwin2.0-Clean2Random** fine-tunes on Clean and evaluates Clean as in-domain (ID) and Randomized as out-of-domain (OOD), exposing transfer; **RoboTwin2.0-Full** fine-tunes on the full RoboTwin2.0 training set and reports the mean success rate over both conditions.

Embodied Pretraining Data Mixture. We compare supervised fine-tuning from scratch with three pretraining strategies under an identical 600-hour data budget, drawing egocentric human video from EgoDex (Hoque et al., 2025) and real-robot manipulation trajectories from RoboCOIN (Wu et al., 2025). *Robot-only* spends the full 600-hour budget on robot data; the two mixed variants combine 350 hours of egocentric data with 250 hours of robot data, either in two stages (*ego then robot*) or jointly in one stage (*ego + robot co-train*). All four variants then undergo identical downstream fine-tuning.

Pretraining Primarily Improves OOD Generalization. As shown in Figure 6, embodied pretraining yields modest gains for in-domain performance, but yields strong performance gains in OOD evaluation. Embodied pretraining therefore contributes mainly knowledge that transfers beyond the downstream training distribution, rather than better fitting an already saturated ID benchmark.

Robot and Egocentric Data Contribute Different Strengths. Robot-only pretraining yields the strongest ID performance, while both mixed strategies generalize better OOD. This trade-off is consistent with the insight of robot trajectories strengthening executable action grounding and egocentric video broadening the visual and interaction distribution.

Absorbing Egocentric Videos: Sequential versus Co-Training. Sequential training and one-stage co-training performance are nearly matched in both in-domain and OOD evaluation, indicating that using both sources matters more than their precise ordering. Co-training is marginally strongest overall and removes the extra curriculum transition, so we adopt it as the practical default.

The from-scratch ablation establishes that the action stream must see the world stream, but leaves one-way and mutual visibility nearly tied. We repeat this comparison after cross-domain pretraining under both RoboTwin2.0-Clean2Random and RoboTwin2.0-Full.

Mutual Visibility Becomes Preferable with Embodied Pretraining. Without embodied pretraining, RoboTwin2.0-Full slightly favors one-way visibility; after pretraining, however, the same protocol favors Mutual. RoboTwin2.0-Clean2Random shows the same reversal in both ID and OOD, with comparable gains across the two splits (Figure 7). The reversal is therefore neither an artifact of domain shift nor of the evaluation protocol: pretraining turns the world–action interaction into a genuinely bidirectional exchange, in which the predicted future frames provide visual guidance for action generation, while the predicted actions in turn inform the synthesis of the manipulator's motion in those frames. Without embodied pretraining, data scarcity likely prevents the two streams from reliably establishing such correspondences; the far more abundant pretraining data closes this gap, and Mutual accordingly realizes its advantage once embodied pretraining is in place. We carry Mutual into the final recipe.

E OpenWAM-α: Implementation Details and Extended Analyses

This section provides the complete specification of OpenWAM-α behind Section 4: its architecture, training, and deployment details, the pretraining data mixture and curation pipeline, the full analysis of its LIBERO-Plus results, and the per-task simulation and real-robot results.

E.1 Architecture Details

OpenWAM-α adopts the architecture that **OpenWAM-Study** converges to (Section 3), assembled from the modules of Appendix C.1 as the composition $C(\mathcal{E}, \mathcal{S}, \mathcal{M})$: the frozen Wan2.2-VAE as the visual encoder $\mathcal{E}$; the pretrained Wan2.2-TI2V-5B DiT (Wan et al., 2025) executing the world stream and a dedicated 1B-parameter ActionDiT executing the action stream, coupled through joint self-attention, as the stream backbones $\mathcal{S} = \{\mathcal{W}, \mathcal{A}\}$; and the *mutual* mode with first-frame-causal intra-video attention as the visibility mask $\mathcal{M}$. Conditioned on the current observation $\mathbf{o}_1$ (all camera views tiled into one canvas), the language instruction ℓ, and the proprioceptive state $\mathbf{q} \in \mathbb{R}^{80}$ expressed in the unified action space, OpenWAM-α jointly denoises the future frames $\mathbf{o}_{2:T}$ of a T-frame video window $\mathbf{o}_{1:T}$ in latent space, together with a continuous action chunk $\mathbf{a} = \mathbf{a}_{1:H} \in \mathbb{R}^{H\times 80}$ (Zhao et al., 2023) (Figure 8a).

Tokenization and Context. The frozen Wan2.2-VAE encodes $\mathbf{o}_{1:T}$ causally – the first frame alone, subsequent frames in groups of four – into $T' = 1 + (T-1)/4$ latent frames $\mathbf{z}$, so the first latent frame remains a clean anchor of the present. A $(1, 2, 2)$ patch embedding flattens the latent video into world-stream tokens of width 3072 carrying 3D RoPE over the (frame, height, width) grid; each of the H noised action steps is linearly embedded into one action-stream token of width 1024 carrying 1D RoPE over the chunk index. The frozen umT5 encoder maps ℓ into a 4096-dimensional context, a linear projection appends $\mathbf{q}$ as one additional context token, and both streams consume the resulting context $\mathbf{c}$ through their own per-block cross-attention.

Stream Bridging and Prediction. All 30 paired layers of the two backbones act as bridge layers, where stream-owned projections map the two residual widths into a shared attention space of 24 heads × 128 dimensions; under the configured $\mathcal{M}$, the two streams read each other freely while the clean first-frame rows attend to neither noised future frames nor actions. AdaLN injects each stream's own denoising timestep – t_v token-wise into the world stream, with first-frame tokens pinned to the clean endpoint $t_v = 1$, and t_a into the action stream – and linear heads decode both final states into velocities, realizing the joint forward pass of Appendix C.2 at any joint noise state (t_v, t_a); how training and inference each cover this plane is specified in Appendices E.2 and E.3.

E.2 Co-Training Setup and Objective

Co-Training Setup. Following the one-stage co-training strategy identified in Section 3.3, OpenWAM-α is trained end to end in a single stage on the ego + robot mixture of Appendix E.4. The video DiT (initialized from the pretrained Wan2.2-TI2V-5B weights), the ActionDiT, and the proprioception encoder are all updated; only the umT5 text encoder and the Wan2.2-VAE remain frozen.

Unified Action Supervision. Heterogeneous embodiments meet in the unified action space of Appendix C.4: an 80-dimensional vector with fixed slot semantics, comprising two mirrored 34-D arm blocks – end-effector position (3), 6D rotation (6), gripper (1), and dexterous hand (24) – followed by 12 slots reserved for embodiment-specific channels (Figure 8c). Each dataset scatters its native action and state into these slots, the validity mask $\mathbf{m}$ of Equation (2) confines action supervision to the populated coordinates, and the robot state at the start of the window supplies the proprioception token in $\mathbf{c}$.

Objective and Timestep Sampling. OpenWAM-α is trained with the joint flow-matching objective of Equation (2) (Black et al., 2024), instantiated with $\lambda_v = \lambda_a = 1$ and a bell-shaped timestep weight $w(\cdot)$ peaked at intermediate noise levels. The per-stream timesteps are drawn independently as $t_v = 1 - f_\rho(u_v)$ and $t_a = 1 - f_\rho(u_a)$ with $u_v, u_a \sim \mathcal{U}[0, 1]$, where the timestep warp f_ρ of Equation (3) is applied identically to both streams with $\rho = 5$ to bias sampling toward high noise.

E.3 Deployment Details

Synchronized Denoising. At test time, OpenWAM-α follows the synchronized schedule selected in Section 3.2: both streams advance in lockstep along the diagonal of the joint noise plane, realized on the training warp as $t^i = 1 - f_\rho(1 - i/N)$ with $\rho = 5$ and $N = 10$ steps, so neither stream leads. Denoising starts from Gaussian noise – with the first latent frame clamped to the encoding of the current observation and re-pinned after every step – and each step performs one joint forward pass

and the coupled Euler update of Appendix C.3, so the action chunk is refined against progressively cleaner world features, and vice versa, at all 30 bridge layers. The finished chunk returns to each robot's native action space through the inverse map of Equation (4).

Inference Mode. All benchmark evaluations that follow, in simulation and in the real world alike, use synchronous inference: whenever the action buffer empties, execution pauses until the model predicts a fresh chunk from the current observation, and the robot then executes that chunk exactly as predicted. The reported results therefore reflect the model's own performance in the most direct way.

Inference Acceleration. OpenWAM-α is served through the acceleration stack of Appendix C.3: the joint denoising loop is compiled into a fixed-shape graph replayed under CUDA graphs, stable velocity predictions are reused across adjacent steps, prompt embeddings are cached, and no VAE decode runs in the control path. The N-step loop completes in roughly 170 ms per chunk on an RTX 5090, well within real-time control budgets.

E.4 Pretraining Data Mixture and Curation

OpenWAM-α is pretrained on multi-domain data drawn from five sources spanning three data types: egocentric human data, real-world robot data, and synthetic robot data. From a raw pool of 1.33B frames (≈14,300 hours), we construct a training set of 518M frames (6,369 hours) through curation and per-source subsampling (Table 3). This section describes how the mixture is composed (Appendix E.4.1) and how each source is cleaned (Appendix E.4.2). We further provide the pretraining-stage hyperparameter configuration and other relevant details in Appendix F.

Table 3: **The OpenWAM-α pretraining data.** *#Emb.* counts each source's distinct embodiments. *Task Coverage* marks the manipulation settings each source spans. *Full* reports each source's raw size before processing, while *Curated + Sampled* reports the data actually used for training, after cleaning (Appendix E.4.2) and per-source whole-episode subsampling (Appendix E.4.1); *Share* is each source's actual per-epoch sample share under proportional sampling.

				Task Coverage				Full		Curated + Sampled		
Source	**Type**	**#Emb.**	**FPS**	Single	Bimanual	Mobile	Dexterous	Frames (M)	Hours	Frames (M)	Hours	Share (%)
Egocentric data (ours)	Human video	1	30	*in-the-wild human manipulation*				744.9	6,897	155.7	1,442	30.1
AgiBotWorld-Beta (Bu et al., 2025)	Real robot	1	15		✓	✓	✓	124.5	2,306	96.9	1,794	18.6
RoboCOIN (Wu et al., 2025)	Real robot	15	30		✓	✓	✓	104.5	956	74.1	686	14.3
DROID (Khazatsky et al., 2024)	Real robot	1	10	✓				46.3	1,285	36.3	1,007	7.0
InternData-A1 (Tian et al., 2025)	Simulation	4	30	✓	✓			313.7	2,904	155.5	1,440	30.0
Total		21 robot + human						**1,333.9**	**14,348**	**518.5**	**6,369**	**100.0**

E.4.1 Pretraining Data Mixture

Data Composition. Following the co-training recipe of Section 3.3, the mixture combines three complementary data types. Egocentric human data comes from a dataset we carefully constructed for manipulation-centric world modeling – 71.6K long-form first-person recordings of 0.25–6 minutes each, covering 3,006 everyday manipulation tasks; it supplies broad visual and interaction diversity but carries no robot action labels, so its action and proprioception channels remain fully masked and it supervises only the world stream. Real-world robot data (AgiBotWorld-Beta (Bu et al., 2025), RoboCOIN (Wu et al., 2025), and DROID (Khazatsky et al., 2024)) grounds the action stream with executable trajectories across 17 physical platforms, while also providing the most faithful visual observations of robots interacting with the physical world. Synthetic robot data (InternData-A1 (Tian et al., 2025)) further broadens the coverage of robot data, encompassing a more comprehensive range of single-arm and bimanual manipulation skills under diverse environmental variations.

Data Budget and Sampling. Considering the compute resources and time cost of pretraining, each source is subsampled under a per-source hour budget. The budgets are derived from frame-based targets – the egocentric and synthetic sources each contribute 30% of the total training frames, and the remaining 40% is divided among the three real-robot sources in proportion to their curated valid-frame counts – so that sources with different native frame rates are balanced by the quantity of data the model actually consumes. Within each source, the hour budget is water-filled across its constituent sub-datasets, and whole episodes are subsampled from the curated pool under a fixed

seed until the budget is met. Training then draws samples proportionally to the actual per-source counts, so every retained sample is visited exactly once per epoch.

E.4.2 Data Curation

Aggregating data across different sources and embodiments introduces heterogeneous defects in both the visual and the signal channel. Our cleaning protocol is informed by the data-cleaning pipeline of Qwen-RobotManip (Yuan et al., 2026a), supplemented with rules for the failure modes we observe in the collected sources, and operates at two levels: vision-level cleaning shared by all sources, and signal-level cleaning specific to robot data.

Vision-Level Cleaning. All sources first pass a uniform visual-quality screen that removes undecodable video, frozen or duplicated frames, black, white, and solid-color frames, over- and under-exposure and exposure flicker, blurred frames, and abrupt visual jumps. Egocentric data further exhibits one failure mode of its own: segments in which the hands leave the field of view carry no manipulation signal and are removed; recordings with empty or invalid language annotations are likewise discarded.

Signal-Level Cleaning. Robot data additionally carries state and action channels, which are cleaned in four steps:

1. **Signal integrity.** An episode is discarded outright when its recorded end-effector state fails to track the commanded actions (amplitude ratio $\geq 3\times$ with per-axis correlation < 0.5), or when video–signal misalignment affects more than 2% of frames.
2. **State-first idle detection.** The state channel serves as the primary criterion for idle footage: leading and trailing segments whose state is static – detected with per-robot motion thresholds calibrated from the p99.5 of single-frame deltas, and confirmed when average end-effector translation and geodesic rotation rates fall below 2 cm/s and 5 °/s – are trimmed, whereas mid-episode pauses are never cut, since cutting them would splice temporally non-adjacent frames. State discontinuities such as jerk and spike outliers are additionally screened with robust median–MAD thresholds.
3. **Visual cross-checking.** When the state does move, it is cross-checked against the visuals: apparent state motion under which every camera view remains visually static is attributed to sensor jitter and trimmed as well, whereas a genuinely moving arm observed by a frozen camera marks a capture defect and the episode is removed.
4. **Episode-level deletion.** An episode that loses more than 70% of its frames to the steps above, or whose video is frozen for 90% or more of its length, is dropped entirely.

E.5 Why Does OpenWAM-α Underperform on LIBERO-Plus?

The unexpected exception is the single-arm benchmark **LIBERO-Plus**, where the scores of OpenWAM-α fall markedly below its standing elsewhere, as shown in Table 4. The per-perturbation breakdown of **LIBERO-Plus** is telling: the losses concentrate under the *camera* and *noise* perturbations, with visible deficits under the *background* and *layout* perturbations as well. On the very same leaderboard, however, ABot-M0.5, ImageWAM, and Being-H0.7 — all WAMs themselves — perform strongly, with scores approaching the state of the art. We therefore compare OpenWAM-α against these models along two axes, pretraining data and architecture, to identify the underlying causes.

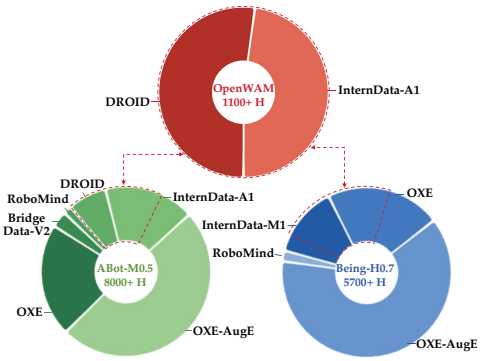

Figure 15: **Single-arm pretraining data of ABot-M0.5, Being-H0.7, and OpenWAM-α.** Dashed lines mark the single-arm data of OpenWAM-α, only a small fraction of the baselines'.

The Data Perspective. Figure 15 contrasts the single-arm portion of the pretraining data of ABot-M0.5 and Being-H0.7 with that of OpenWAM-α. Both baselines pretrain on far larger and more varied single-arm collections, spanning diverse embodiments, scenes, and camera viewpoints, so during

Table 4: **Evaluation Results on LIBERO-Plus.** Bold denotes best values, underline second best.

	Camera	Robot	Language	Light	Background	Noise	Layout	Avg
VLA								
π_0 (Black et al., 2024)	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
OpenVLA-OFT (Kim et al., 2025)	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
StarVLA (Community, 2026)	52.5	49.8	88.5	95.7	95.7	73.0	76.9	74.1
ABot-M0 (Yang et al., 2026b)	60.4	67.9	86.4	96.2	91.6	86.4	82.6	80.5
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	78.4	73.6	80.8	96.2	94.1	89.0	84.5	84.4
ACoT-VLA (Zhong et al., 2026)	72.6	<u>82.6</u>	87.5	97.7	<u>96.5</u>	87.8	<u>88.1</u>	<u>86.6</u>
QWEN-ROBOTMANIP (Yuan et al., 2026a)	**87.2**	75.5	85.6	96.6	**97.7**	**97.7**	87.3	**89.0**
WAM								
Fast-WAM (Yuan et al., 2026b)	16.4	44.5	68.9	78.2	53.7	37.7	60.7	51.5
Being-H0.7 (Luo et al., 2026b)	<u>82.0</u>	59.0	82.8	<u>97.8</u>	90.0	93.5	**88.5**	82.1
Cosmos-Policy (Kim et al., 2026b)	75.8	63.3	81.7	96.5	88.9	92.7	82.2	82.2
ImageWAM (Zhang et al., 2026c)	80.8	50.3	**91.4**	**98.1**	85.5	<u>93.8</u>	80.5	83.1
ABot-M0.5 (Chen et al., 2026a)	70.5	**87.4**	<u>88.6</u>	94.0	89.7	75.5	85.2	83.4
OpenWAM-α	33.8	76.1	88.0	97.0	87.1	39.8	77.5	69.2

pretraining they have already seen visual information and world knowledge close to the LIBERO-Plus test scenes — in viewpoint and noise as much as in background and layout. In contrast, the single-arm data of OpenWAM-α (Table 3) is far smaller in both volume and variety, drawing on only two sources: DROID, collected on a fixed single-arm platform with fixed camera viewpoints, and the single-arm portion of the synthetic InternData-A1. With such limited single-arm coverage, the model receives far less single-arm world knowledge, and its single-arm generalization suffers accordingly on the OOD perturbations of LIBERO-Plus. The converse also holds: the OpenWAM-α mixture is rich in egocentric, bimanual, and dexterous-hand data, and the model is correspondingly strong on the bimanual and dexterous-hand benchmarks.

The Architecture Perspective. Fast-WAM, ABot-M0.5, and OpenWAM-α share one core prediction target — a pixel-level video of the future over a temporal horizon, with the intermediate latents produced by the reconstructive Wan2.2-VAE. On LIBERO-Plus, all three exhibit the same signature: the scores under the *camera* and *noise* perturbations fall clearly below those under the other perturbations, and only ABot-M0.5, backed by its pretraining data, recovers much of the loss relative to Fast-WAM and OpenWAM-α. Pixel-level information is evidently acutely sensitive to camera and noise perturbations — an inherent limitation of pixel-level prediction architectures that only large-scale pretraining can compensate. Two further baselines corroborate this reading. ImageWAM, although not pretrained, remains conspicuously strong on LIBERO-Plus: its prediction target is also a pixel-level latent, but it predicts only a single future frame of the current observation — closer to an edit than a rollout — so no error accumulates across frames and the impact of future-pixel prediction on the camera and noise scores shrinks accordingly. Being-H0.7, in turn, encodes observations with V-JEPA 2.1: its intermediate latents remain temporal (several frames are encoded jointly), yet they are semantic-level features rather than pixel reconstructions, which makes the model markedly more robust to the camera and noise perturbations. Unlike LIBERO-Plus, the OOD designs of the other benchmarks impose no deliberate camera or noise disturbance, so OpenWAM-α remains highly competitive there; on LIBERO-Plus, the camera and noise disturbances compound the single-arm data deficit above, and the performance of OpenWAM-α inevitably degrades.

E.6 REAL-ROBOT EXPERIMENTS AND RESULTS

To further examine OpenWAM-α beyond simulation and validate both its general capability and its generalization, we conduct comprehensive real-robot evaluations across three embodiments — single-arm, bimanual, and dexterous-hand — with the experimental setups shown in Figure 16. Specifically:

- **Single-arm experiments** are conducted on the Franka-Research-3 platform and cover three task families — stacking, pick-and-place, and hanging — probing the model's basic and fine-grained manipulation capabilities. Performance is measured by task success rate (SR).

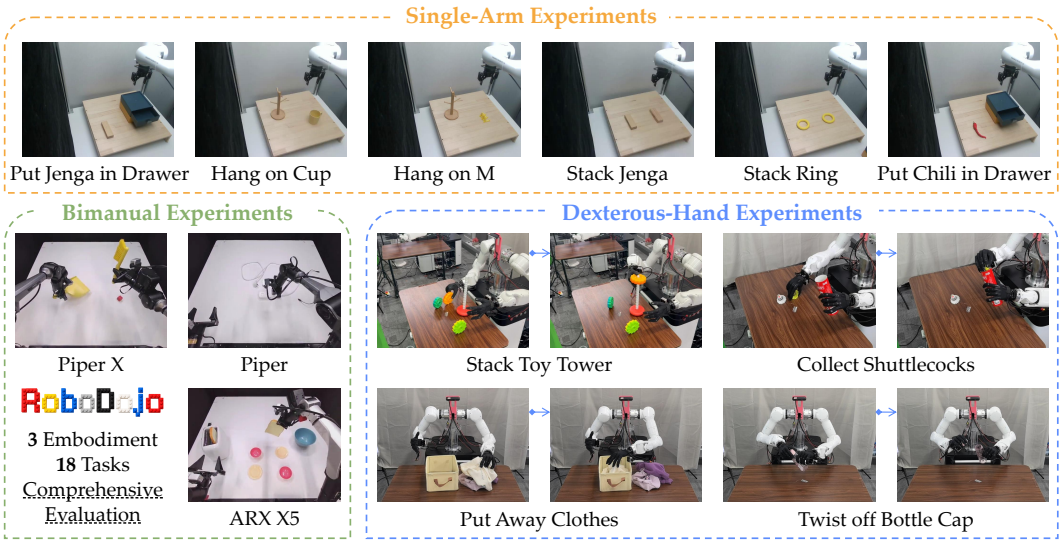

Figure 16: **Real-robot experimental setups across three embodiments.** *Top*: the six single-arm tasks on the Franka-Research-3 platform. *Bottom left*: the three bimanual embodiments of the RoboDojo real-world track (Piper X, Piper, and ARX X5), covering 18 tasks in total. *Bottom right*: the four dexterous-hand tasks on the Wuji-hand and Tianji-arm platform, each illustrated by key intermediate stages of its execution.

- **Bimanual experiments** are conducted on the official RoboDojo real-robot platform, spanning three embodiments (ARX X5, Piper, and Piper X); following the RoboDojo task taxonomy, the evaluation covers generalization, precision, long-horizon, memory, and open tasks, assessing the model comprehensively. Performance is measured by SR and Progress Score.
- **Dexterous-hand experiments** are conducted on a platform pairing the Wuji dexterous hand with the Tianji robotic arm — an embodiment and action space absent from the OpenWAM-α pretraining mixture — and cover bimanual-interactive, long-horizon, and fine manipulation tasks, probing how well the model adapts and generalizes to unseen embodiments and unseen action dimensions. Performance is measured by SR and Progress Score.

The task setups and evaluation protocols of each embodiment are documented in Appendix G, and the SFT configurations used for these experiments are provided in Appendix F.2.

Table 5, Table 6, and Table 7 report the detailed scores of the single-arm, RoboDojo bimanual, and dexterous-hand experiments, respectively.

Table 5: **Evaluation Results on Single-Arm Real-Robot Tasks.** Bold denotes best values, underline second best.

	Stack Jenga	Stack Ring	Put Chili in Drawer	Put Jenga in Drawer	Hang on M	Hang on Cup	Avg
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	13/20 (65%)	7/20 (35%)	14/20 (70%)	15/20 (75%)	7/20 (35%)	10/20 (50%)	66/120 (55.0%)
LingBot-VA (Li et al., 2026b)	16/20 (80%)	**15/20 (75%)**	17/20 (85%)	17/20 (85%)	**13/20 (65%)**	15/20 (75%)	93/120 (77.5%)
OpenWAM-α	**17/20 (85%)**	12/20 (60%)	**20/20 (100%)**	**20/20 (100%)**	**13/20 (65%)**	**17/20 (85%)**	**99/120 (82.5%)**

On the single-arm platform (Table 5), OpenWAM-α clearly leads both LingBot-VA, a representative WAM, and $\pi_{0.5}$, a representative VLA, on the majority of tasks, and attains the best average success rate, providing initial evidence of its general and fine-grained manipulation capabilities. We then turn to RoboDojo-Real (Table 6), a real-robot benchmark that comprehensively evaluates generalist manipulation policies across three bimanual embodiments and a wide range of task dimensions. OpenWAM-α tops the leaderboard: it remains consistently strong across the embodiments and their tasks, and reaches the state of the art on the great majority of them, corroborating the generality and robustness of the model in the real world.

Table 6: **Evaluation Results on the Bimanual RoboDojo Real-World Track.** Each cell reports Score / SR (%). Bold denotes best values, underline second best. Rows are shaded by embodiment.

Policy	Embodiment	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Emb. Avg.	Overall Avg.
X-VLA (Zheng et al., 2026b)	ARX X5	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	2.0 / 0.0	20.7 / 10.0	18.0 / 0.0	6.8 / 1.7	7.6 / 3.3
	Piper	0.0 / 0.0	5.3 / 0.0	0.0 / 0.0	0.0 / 0.0	53.0 / 50.0	37.0 / 0.0	15.9 / 8.3	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.7 / 0.0	0.1 / 0.0	
Xiaomi-Robotics-0 (Cai et al., 2026c)	ARX X5	24.0 / **20.0**	0.0 / 0.0	3.0 / 0.0	4.0 / 0.0	40.0 / 20.0	19.0 / 10.0	15.0 / 8.3	7.9 / 3.9
	Piper	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	23.0 / 20.0	22.7 / 0.0	7.6 / 3.3	
	Piper X	0.0 / 0.0	0.7 / 0.0	4.0 / 0.0	0.0 / 0.0	0.0 / 0.0	2.7 / 0.0	1.2 / 0.0	
GalaxeaVLA (G0) (Jiang et al., 2025)	ARX X5	1.0 / 0.0	0.0 / 0.0	3.0 / 0.0	6.0 / 0.0	0.0 / 0.0	10.3 / 0.0	3.4 / 0.0	9.0 / 4.4
	Piper	0.0 / 0.0	32.7 / 10.0	13.3 / 10.0	0.0 / 0.0	56.0 / 50.0	30.0 / 10.0	22.0 / 13.3	
	Piper X	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	4.0 / 0.0	0.0 / 0.0	6.0 / 0.0	1.7 / 0.0	
InternVLA-A1 (Cai et al., 2026b)	ARX X5	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	48.0 / 20.0	12.0 / 0.0	10.0 / 3.3	12.0 / 7.2
	Piper	0.0 / 0.0	7.3 / 0.0	0.0 / 0.0	0.0 / 0.0	73.0 / 70.0	59.0 / 40.0	23.2 / 18.3	
	Piper X	0.0 / 0.0	0.0 / 0.0	4.0 / 0.0	2.0 / 0.0	0.0 / 0.0	10.0 / 0.0	2.7 / 0.0	
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	ARX X5	24.6 / **20.0**	**1.8** / 0.0	**25.8 / 10.0**	47.0 / **20.0**	40.0 / 20.0	26.8 / 10.0	27.7 / 13.3	22.9 / 12.8
	Piper	**10.0 / 10.0**	28.0 / 0.0	10.0 / 10.0	0.0 / 0.0	72.0 / 60.0	72.0 / **50.0**	32.0 / 21.7	
	Piper X	0.0 / 0.0	14.8 / **10.0**	0.0 / 0.0	29.5 / 10.0	7.5 / 0.0	3.0 / 0.0	9.1 / 3.3	
OpenWAM-α	ARX X5	**31.0** / 0.0	0.0 / 0.0	13.0 / 0.0	**52.0 / 20.0**	**100.0 / 100.0**	**38.0 / 20.0**	**39.0 / 23.3**	**37.6 / 24.4**
	Piper	8.3 / 0.0	**60.0 / 40.0**	**36.7 / 30.0**	0.0 / 0.0	**100.0 / 100.0**	**75.0 / 50.0**	**46.7 / 36.7**	
	Piper X	0.0 / 0.0	**28.0 / 10.0**	**18.0** / 0.0	**52.3 / 20.0**	**40.0 / 40.0**	**24.7 / 10.0**	**27.2 / 13.3**	

Task order. **ARX X5**: `cover_blocks`, `make_bread`, `make_food`, `pack_and_pour_fruit`, `store_in_safe`, `insert_tubes`. **Piper**: `stack_and_cover_blocks`, `fill_pen_holder`, `put_objects_into_basket`, `insert_charger`, `stack_bowls`, `stand_up_bottles`. **Piper X**: `classify_objects`, `disassemble_LEGO`, `hang_mugs`, `pack_objects_into_backpack`, `sweep_blocks`, `cap_pen`.

Table 7: **Evaluation Results on Dexterous-Hand Real-Robot Tasks.** Each cell reports Score / SR (%); the OOD column aggregates all variation settings of its task. Bold denotes best values.

	Stack Toy Tower		Collect Shuttlecocks	
Method	**ID**	**OOD**	**ID**	**OOD**
$\pi_{0.5}$	16/30 (53.3) / 1/10 (10%)	12/30 (40.0) / 1/10 (10%)	11/35 (31.4) / 2/10 (20%)	17/54 (31.5) / 3/15 (20%)
OpenWAM-α	**21/30 (70.0) / 4/10 (40%)**	**17/30 (56.7) / 3/10 (30%)**	**29/35 (82.9) / 6/10 (60%)**	**30/57 (52.6) / 4/15 (26.7%)**

	Put Away Clothes		Twist off Bottle Cap	
Method	**ID**	**OOD**	**ID**	**OOD**
$\pi_{0.5}$	26/30 (86.7) / 6/10 (60%)	45/60 (75.0) / 9/20 (45%)	8/10 (80.0) / 3/10 (30%)	13/20 (65.0) / 6/20 (30%)
OpenWAM-α	**30/30 (100.0) / 10/10 (100%)**	**55/60 (91.7) / 17/20 (85%)**	**10/10 (100.0) / 7/10 (70%)**	**18/20 (90.0) / 16/20 (80%)**

To further probe the extensibility and generalization of OpenWAM-α, we fine-tune the pretrained model on four dexterous manipulation tasks built on the Wuji-hand and Tianji-arm platform and test it under both in-domain and out-of-domain setups (Table 7). Neither the platform nor its action space — a 9-D end-effector pose combined with 21 dexterous-hand degrees of freedom — ever appears in the OpenWAM-α pretraining mixture; nevertheless, OpenWAM-α outperforms $\pi_{0.5}$ by a clear margin across all tasks and setups, demonstrating that the model adapts reliably and stably to an entirely unseen embodiment.

Together, these experiments assess OpenWAM-α across three embodiment types and a broad spectrum of real-world manipulation tasks. The results show that OpenWAM-α performs strongly in every setting and stands on par with today's leading models, indicating that it can serve as a strong baseline for further development and comparison by the community.

F Training Details

This section documents the optimization and data-loading configurations used to train and adapt OpenWAM-α. We organize the details into two stages: multi-domain pretraining and dataset-specific supervised fine-tuning (SFT) during post-training.

Table 8: **Pretraining configuration for OpenWAM-α.**

Configuration	Value
Compute	16 nodes (128 NVIDIA H200 GPUs)
Training time	$\approx$ 7 days
Optimizer	AdamW
Batch size	3,072 (24 per GPU)
Learning rate	1×10^{-4}
LR schedule	Cosine; 5% warmup; minimum ratio 0.01
Weight decay	0.01
Optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$
Training iterations	155,862 (1 epoch)
Gradient clipping	Global norm 1.0
Model precision	bfloat16
Distributed training	DeepSpeed ZeRO Stage 2
Input clip	33 frames; video stride 4; window stride 1
Image resolution	384×320
Multi-view input	Enabled
Image augmentation	`ColorJitter`(0.2, 0.2, 0.2, 0.0)
Flow shifts	Video/action: $5.0/5.0$
Loss weights	$\lambda_v = 1.0$, $\lambda_a = 1.0$
Unified control space	80-D action; 80-D proprioceptive state

Table 9: **Dataset-specific SFT configuration for OpenWAM-α.** Settings not listed are identical to pretraining (Table 8); "–" denotes no image augmentation.

Benchmark	Batch size	Training Epochs / Steps	Augmentation
Simulation benchmarks			
LIBERO	256	10 epochs (10,690 steps)	–
VLABench	196	6k steps	ColorJitter
RoboTwin2.0-Full	256	5 epochs (118,655 steps)	–
RoboTwin2.0-Clean2Random	256	5 epochs (10,740 steps)	ColorJitter
RoboDojo	256	60k steps	ColorJitter
RoboCasa365	1,024	60k steps	ColorJitter
EBench	256	100k steps	ColorJitter
RoboCasa-GR1	256	100k steps	ColorJitter
Real-robot experiments			
Single-arm (Franka-Research-3)	256	10 epochs (9,860 steps)	–
Bimanual (RoboDojo real-world track)	256	30k steps	ColorJitter
Dexterous hand (Wuji + Tianji)	256	5 epochs (10,925 steps)	–

F.1 Pretraining Configuration

The key hyperparameters used for multi-domain pretraining are summarized in Table 8. Pretraining uses 16 nodes with eight NVIDIA H200 GPUs per node (128 GPUs in total) and takes approximately seven days. With 24 clips per GPU and no gradient accumulation, the global batch size is 3,072 clips per optimizer step. The source-level data budgets and realized mixture proportions are reported separately in Table 3.

F.2 Dataset-Specific SFT Configuration

All downstream models are initialized from the same pretrained OpenWAM-α checkpoint. Supervised fine-tuning keeps the pretraining configuration of Table 8 unchanged and differs only in the three benchmark-dependent settings summarized in Table 9: the global batch size, the number of

training epochs or steps, and whether image augmentation is applied. Training length is given in epochs over the fine-tuning set, with the corresponding number of optimizer steps in parentheses, or directly in optimizer steps where no epoch-based schedule was used. Image augmentation, where enabled, is the same `ColorJitter`(0.2, 0.2, 0.2, 0.0) used in pretraining. LIBERO-Plus is evaluated with the LIBERO checkpoint without further fine-tuning.

G Real-World Evaluation Protocols

This section details the real-world evaluation of Section 4.3: for each embodiment, we document the task setup and the corresponding evaluation protocol.

G.1 Single-Arm Real-Robot Experiments

Task Setup. We evaluate single-arm policies on the Franka-Research-3 platform using six real-world tabletop tasks, covering stacking, hanging, and drawer manipulation. The tasks use a Franka-Research-3 arm with a parallel gripper and RGB observation cameras, as shown in Figure 17. Each task is specified by a natural-language instruction and instantiated with a fixed physical scene: stacking tasks place two target objects on the tabletop, hanging tasks place the object and shelf in the workspace, and drawer tasks place the object on the table next to an upper drawer.

Table 10: Task instructions used in the single-arm real-robot evaluation.

Task	Instruction
Stack Ring	Pick the yellow ring on the left side, stack it on the other ring.
Stack Jenga	Pick the jenga on the left side, stack it on the other jenga.
Hang on Cup	Pick the cup on the table, hang it on the shelf.
Hang on M	Pick the M-shaped object on the table, hang it on the shelf.
Put Chili in Drawer	Pick the chili on the table, put it into the drawer, then push the upper drawer closed.
Put Jenga in Drawer	Pick up the jenga block on the table, put it into the drawer, then push the upper drawer closed.



Figure 17: Single-arm real-robot setup on the Franka-Research-3 platform. The workspace contains the drawer, stacking objects, and hanging fixtures used across the six tasks, while the robot is equipped with an Intel RealSense camera and a Robotiq parallel gripper for closed-loop execution.

The corresponding execution sequences are visualized in Figure 18, where each row contains six frames uniformly sampled from one rollout video of the task.

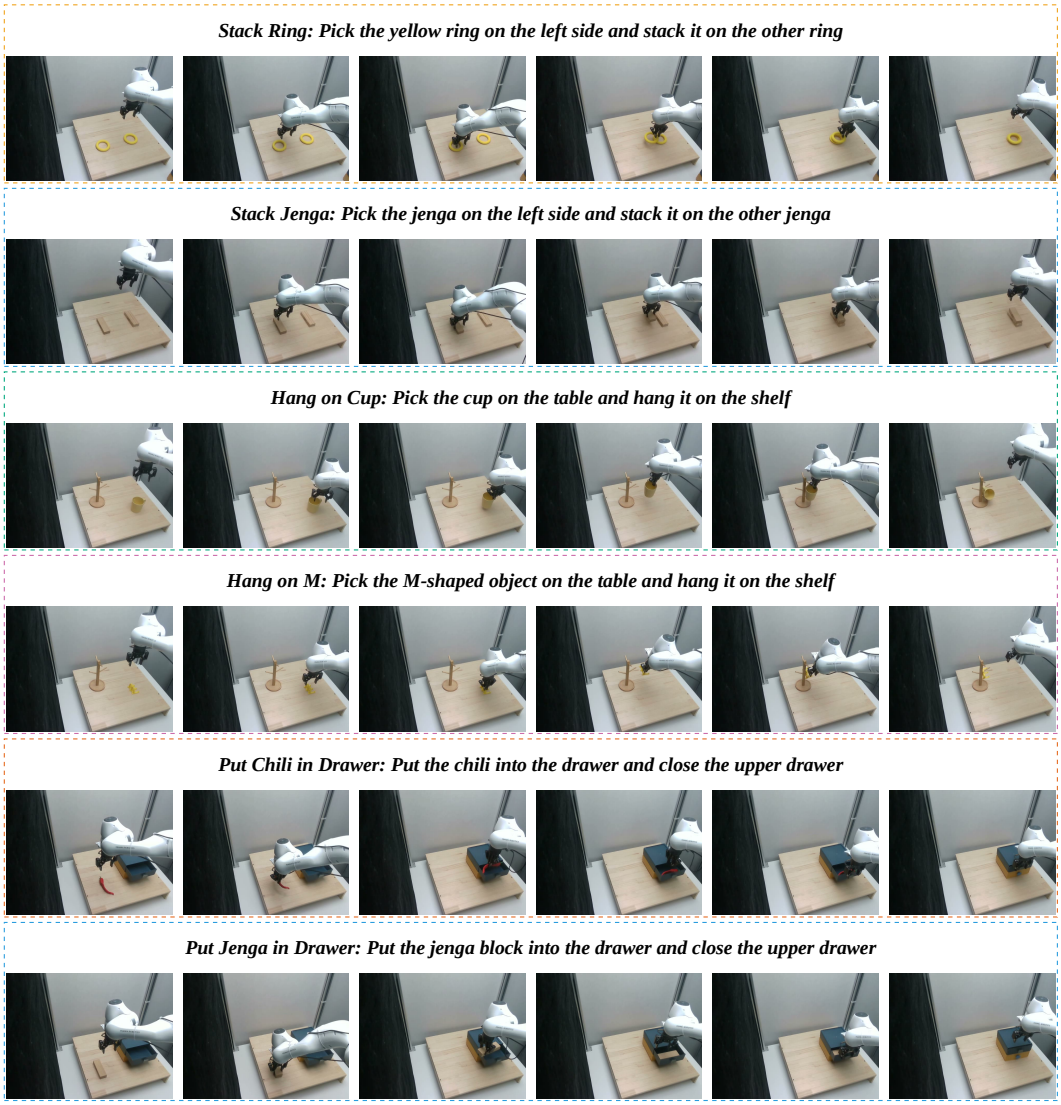

Figure 18: Single-arm task execution sequences on the Franka-Research-3 platform. Each row shows one task, with six frames uniformly sampled from the corresponding left-view rollout video.

Evaluation Protocol. For each single-arm task, we fine-tune the policy with 100 task-specific real-robot demonstrations. After fine-tuning, each task is evaluated over 20 independent real-world trials, and we report the success rate as the number of successful trials out of 20 in Table 5. A stacking trial succeeds only if the left object is picked and stably stacked on the target object. A hanging trial succeeds only if the object is picked and remains hanging on the shelf. A drawer trial succeeds only if the object is placed inside the drawer and the upper drawer is pushed closed.

G.2 Dexterous-Hand Real-Robot Experiments

Task Setup. We evaluate dexterous-hand policies on four real-world manipulation tasks: Stack Toy Tower, Collect Shuttlecocks, Twist off Bottle Cap, and Put Away Clothes. Each task is specified by a natural-language instruction that defines the desired manipulation objective. The task instructions are summarized in Table 11.

Table 11: Task instructions used in the real-world evaluation.

Task	Instruction
Stack Toy Tower	Stack the discs onto the tower pole in order from largest to smallest.
Collect Shuttlecocks	Put all the shuttlecocks into the shuttlecock tube.
Twist off Bottle Cap	Twist off the bottle cap.
Put Away Clothes	Pick up the clothes from the pile on the table and put them into the basket.

All experiments are conducted on a physical dexterous-hand platform. The robot receives the task instruction and executes the manipulation autonomously in the corresponding scene. Figure 19 shows the physical robot, dexterous hand, workspace, camera viewpoint, and representative objects used in the evaluation.



Figure 19: Real-world experimental setup. The figure shows the physical dexterous-hand platform, the workspace, the observation camera, and representative objects for the four manipulation tasks.

For each task, we evaluate the policy under an in-distribution (ID) condition and several out-of-distribution (OOD) conditions. The OOD conditions modify one factor at a time, including object identity, object layout, illumination, or background appearance, while preserving the task instruction and the overall manipulation objective. The task execution sequence and the corresponding ID/OOD configurations are illustrated in Figure 20.

Evaluation Protocol. Each task is evaluated over multiple independent trials under both ID and OOD conditions. We report two complementary metrics: the progress score (Score) and the final success rate (SR).

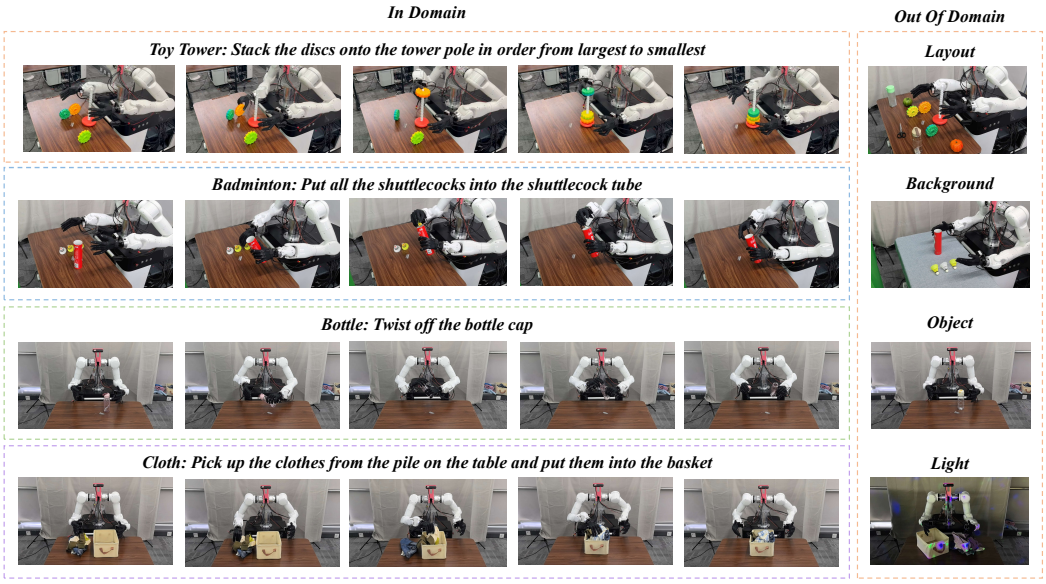

Figure 20: Task execution sequences and evaluation conditions. Each row illustrates the main manipulation stages of one task, while the columns show the corresponding ID scene and OOD variations. OOD conditions include changes in object identity, object layout, illumination, and background appearance.

Final Success Rate. A trial is counted as a final success only when the complete task objective is achieved. The final success rate is computed as

$$S_{\text{final}} = \frac{N_{\text{success}}}{N_{\text{trial}}} \times 100\%,$$

where N_{success} is the number of trials satisfying the complete task criterion and N_{trial} is the total number of valid trials.

Progress Score. The progress score measures the fraction of required manipulation elements that are successfully completed, regardless of whether the final task state is achieved. For trial i, let n_i denote the number of successfully completed elements and m_i denote the total number of elements present in that trial. The progress score is computed as

$$S_{\text{process}} = \frac{\sum_i n_i}{\sum_i m_i} \times 100\%.$$

Task-Specific Criteria.

- **Collect Shuttlecocks.** Each trial contains two to four shuttlecocks on the tabletop. A trial is counted as a final success only when all shuttlecocks are placed into the shuttlecock tube. The progress score is the fraction of shuttlecocks successfully placed into the tube.
- **Stack Toy Tower.** Each trial contains three discs. A trial is counted as a final success only when all three discs are successfully inserted onto the tower pole in descending order of size. The progress score is the fraction of discs successfully inserted.
- **Put Away Clothes.** Each trial contains three to four pieces of clothing. A trial is counted as a final success only when all pieces of clothing are placed into the basket. The progress score is the fraction of clothing items successfully placed into the basket.
- **Twist off Bottle Cap.** A trial is counted as a final success when the bottle cap is fully twisted off and the robot maintains a stable grasp of the bottle or cap. The progress score records whether the cap is successfully twisted off, regardless of whether the final stable grasp is achieved.

G.3 BIMANUAL REAL-ROBOT EXPERIMENTS

We conduct a comprehensive evaluation on all 18 tasks of RoboDojo-Real, using the official real-robot evaluation platform provided by the RoboDojo team and covering its three bimanual embodiments: ARX X5, Piper, and Piper X. The evaluation strictly follows the unified protocol defined by the official evaluation team; detailed documentation of the platform and its task specifications is available on the RoboDojo website (`https://robodojo-benchmark.com/`) and in the accompanying official documentation (`https://robodojo-benchmark.com/doc/real-tasks/`).

H PER-BENCHMARK SIMULATION RESULTS

The tables below report the full per-benchmark scores summarized in Figure 9, except for LIBERO-Plus, whose scores are already reported in Table 4 of the main text. Within each table the baselines are grouped as VLA or WAM; bold denotes the best value and underline the second best.

Table 12: **Evaluation Results on LIBERO.** Bold denotes best values, underline second best.

	Spatial	Object	Goal	Long	Avg
VLA					
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5
π_0 (Black et al., 2024)	98.0	96.8	94.4	88.4	94.4
StarVLA (Community, 2026)	97.8	98.6	96.2	93.8	96.6
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	98.8	98.2	98.0	92.4	96.9
GR00T-N1.6 (NVIDIA et al., 2025)	97.7	98.5	97.5	94.4	97.0
OpenVLA-OFT (Kim et al., 2025)	97.6	98.4	97.9	94.5	97.1
X-VLA (Zheng et al., 2026b)	98.2	98.6	97.8	97.6	98.1
ABot-M0 (Yang et al., 2026b)	98.8	<u>99.8</u>	99.0	96.6	98.6
Being-H0.5 (Luo et al., 2026a)	99.2	99.6	<u>99.4</u>	97.4	98.9
QWEN-ROBOTMANIP (Yuan et al., 2026a)	–	–	–	–	99.2
WAM					
Fast-WAM (Yuan et al., 2026b)	98.2	**100.0**	97.0	95.2	97.6
Motus (Bi et al., 2026)	96.8	<u>99.8</u>	96.6	97.6	97.7
ImageWAM (Zhang et al., 2026c)	97.2	99.2	98.8	<u>98.4</u>	98.4
LingBot-VA (Li et al., 2026b)	98.5	99.6	97.2	**98.5**	98.5
DiT4DiT (Ma et al., 2026)	–	–	–	–	98.6
Being-H0.7 (Luo et al., 2026b)	–	–	–	–	99.2
ABot-M0.5 (Chen et al., 2026a)	**100.0**	<u>99.8</u>	<u>99.4</u>	<u>98.4</u>	**99.4**
OpenWAM-α	<u>99.6</u>	99.6	**99.8**	98.2	<u>99.3</u>

Table 13: **Evaluation Results on VLABench.** Bold denotes best values, underline second best.

	In-dist.			Category			Commonsense			Instruction			Texture			Avg		
	SR	PS	IS	SR	PS	IS	SR	PS	IS	SR	PS	IS	SR	PS	IS	SR	PS	IS
VLA																		
π_0 (Black et al., 2024)	47.0	62.7	67.8	21.2	33.6	44.0	29.1	43.0	54.9	17.3	38.7	58.0	32.2	42.5	50.6	29.4	44.1	55.0
LoHo-Manip (Liu et al., 2026a)	54.0	–	–	23.0	–	–	36.0	–	–	42.0	–	–	39.0	–	–	39.0	–	–
ACoT-VLA (Zhong et al., 2026)	–	66.1	79.8	–	38.9	54.1	–	37.8	52.3	–	39.6	56.8	–	54.6	74.6	–	47.4	63.5
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	65.4	77.8	80.4	38.2	49.7	52.0	43.9	57.3	60.0	48.2	64.2	67.0	44.9	62.3	65.0	48.1	62.3	64.9
ERVLA (Sun et al., 2026)	69.7	81.1	84.2	47.0	61.0	**66.4**	44.0	55.0	57.2	58.0	70.2	73.8	47.4	62.3	70.6	53.2	65.9	70.4
Xiaomi-Robotics-1 (Team et al., 2026b)	75.6	85.0	79.8	**53.0**	**66.6**	**66.4**	48.4	58.3	58.2	55.8	66.8	70.2	**62.6**	**74.9**	74.8	**59.1**	**70.3**	69.9
WAM																		
Bridge-WA (Bai et al., 2026)	78.0	85.8	**85.0**	23.0	28.8	39.0	51.1	64.4	**74.2**	**67.0**	**80.3**	**82.0**	45.0	60.3	**76.0**	52.8	64.0	**71.2**
OpenWAM-α	**83.4**	**87.9**	75.2	38.1	45.9	45.9	**58.0**	**64.8**	57.9	53.8	64.5	66.5	61.4	72.9	71.6	58.9	67.2	63.5

Table 14: **Evaluation Results on RoboTwin2.0-Clean2Random.** Bold denotes best values, underline second best.

	Clean	Randomized	Avg
VLA			
StarVLA (Community, 2026)	46.5	3.2	24.9
GR00T-N1.7 (NVIDIA et al., 2025)	43.6	20.7	32.2
X-VLA (Zheng et al., 2026b)	68.0	20.9	44.5
Spatial Forcing (Li et al., 2026a)	77.2	26.7	52.0
ABot-M0 (Yang et al., 2026b)	70.7	36.0	53.4
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	70.7	46.0	58.4
GigaBrain-0.7 (Team et al., 2026a)	66.8	67.9	67.4
QWEN-ROBOTMANIP (Yuan et al., 2026a)	84.7	**69.4**	**77.1**
WAM			
AHA-WAM (Cai et al., 2026a)	64.3	3.2	33.8
Fast-WAM (Yuan et al., 2026b)	77.8	1.9	39.9
X-WAM (Guo et al., 2026)	70.0	25.8	47.9
4D-WAM (Yang et al., 2026a)	81.5	41.8	61.7
OpenWAM-α	**89.4**	48.7	69.0

Table 15: **Evaluation Results on RoboTwin2.0-Full.** Bold denotes best values, underline second best.

	Clean	Randomized	Avg
VLA			
X-VLA (Zheng et al., 2026b)	72.80	72.84	72.82
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	82.70	76.80	79.75
ABot-M0 (Yang et al., 2026b)	86.06	85.08	85.57
Qwen-VLA (Wang et al., 2026a)	86.10	87.20	86.65
StarVLA (Community, 2026)	88.18	88.32	88.25
Galaxea G0.5 (Liu et al., 2026b)	93.70	92.80	93.25
QWEN-ROBOTMANIP (Yuan et al., 2026a)	93.70	94.00	93.85
WAM			
Motus (Bi et al., 2026)	88.66	87.02	87.84
Fast-WAM (Yuan et al., 2026b)	91.90	91.80	91.85
LingBot-VA (Li et al., 2026b)	92.93	91.55	92.24
ImageWAM (Zhang et al., 2026c)	93.20	93.56	93.38
LingBot-VA 2.0 (Zhang et al., 2026b)	93.80	93.40	93.60
ABot-M0.5 (Chen et al., 2026a)	**94.00**	**94.20**	**94.10**
OpenWAM-α	93.74	93.46	93.60

Table 16: **Evaluation Results on RoboDojo.** Bold denotes best values, underline second best.

	Gen-Std		Gen-Rand		Precision		Long-Horizon		Memory		Open		Avg	
	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score
VLA														
StarVLA-α (Ye et al., 2026b)	5.00	7.54	0.00	0.33	4.33	9.90	6.50	14.15	2.44	3.34	0.58	0.68	3.24	6.40
X-VLA (Zheng et al., 2026b)	12.00	17.90	1.00	3.04	12.00	18.32	9.75	16.53	3.56	4.76	0.50	0.55	6.52	10.13
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	15.00	20.93	1.00	5.82	5.50	12.40	14.67	23.54	4.56	5.78	1.67	1.98	6.91	11.41
Spatial Forcing (Li et al., 2026a)	15.00	21.25	4.00	6.98	10.58	17.33	14.58	23.26	4.11	5.43	1.58	1.78	8.04	12.38
Hy-Embodied-0.5-VLA (Zhang et al., 2026a)	17.00	21.98	0.00	1.57	8.00	13.81	14.92	25.74	12.11	13.37	0.58	0.65	8.80	13.07
Xiaomi-Robotics-1 (Team et al., 2026b)	**28.00**	**35.65**	**6.00**	**11.44**	18.83	26.69	23.67	38.39	6.56	7.81	**3.58**	**3.94**	13.93	20.07
Galaxea G0.5 (Liu et al., 2026b)	20.00	26.74	**6.00**	11.16	**20.42**	**28.25**	**32.25**	**44.12**	7.33	8.61	1.58	1.73	14.88	20.23
DM0.5 (Dexmal, 2026)	18.00	23.49	4.00	8.06	16.75	24.82	19.50	33.70	**47.44**	**47.74**	2.08	2.43	**19.34**	**24.90**
WAM														
Fast-WAM (Yuan et al., 2026b)	2.00	4.33	0.00	0.34	0.00	1.96	5.17	9.14	3.44	3.55	0.42	0.42	2.03	3.48
AHA-WAM (Cai et al., 2026a)	6.00	10.32	0.00	1.26	2.42	5.86	2.67	8.61	2.78	2.97	0.83	0.88	2.39	4.82
GigaWorld-Policy (Ye et al., 2026a)	6.00	10.28	0.00	0.41	1.83	6.15	8.92	15.51	2.22	3.46	0.50	0.54	3.27	6.20
X-WAM (Guo et al., 2026)	5.00	11.24	1.00	3.54	1.83	6.72	9.08	17.47	4.67	6.32	0.25	0.57	3.83	7.69
OpenWAM-α	25.56	33.16	4.11	8.26	9.25	18.45	25.33	34.93	9.11	10.41	1.08	1.41	11.92	17.18

Table 17: **Evaluation Results on EBench.** Bold denotes best values, underline second best.

	Table Top		Simple PnP		Long Horizon		Overall	
	SR	**Score**	**SR**	**Score**	**SR**	**Score**	**SR**	**Score**
VLA								
StarVLA-OFT (Community, 2026)	–	–	–	–	–	–	0.0	0.2
π_0 (Black et al., 2024)	15.7	30.0	35.0	39.0	17.0	41.0	23.6	37.0
X-VLA (Zheng et al., 2026b)	8.6	24.0	50.0	54.0	6.2	25.0	23.7	36.0
InternVLA-A1 (Cai et al., 2026b)	4.3	11.0	43.0	47.0	17.9	46.0	23.9	36.0
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	12.9	32.0	45.0	50.0	18.1	39.0	27.1	41.0
GigaBrain-0.7 (Team et al., 2026a)	–	–	–	–	–	–	33.3	46.0
QWEN-ROBOTMANIP (Yuan et al., 2026a)	**50.0**	**70.0**	56.5	60.0	29.9	55.0	45.6	60.0
WAM								
Fast-WAM (Yuan et al., 2026b)	–	–	–	–	–	–	4.7	7.6
OpenWAM-α	30.0	44.2	**67.5**	**72.0**	**44.3**	**72.6**	**49.4**	**64.7**

Table 18: **Evaluation Results on RoboCasa365.** Bold denotes best values, underline second best.

	Atomic	Comp.-Seen	Comp.-Unseen	Avg
VLA				
Diffusion Policy (Chi et al., 2025)	15.7	0.2	1.3	6.1
π_0 (Black et al., 2024)	36.3	5.2	0.7	15.0
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	39.6	7.1	1.2	16.9
GR00T-N1.5 (NVIDIA et al., 2025)	50.7	14.8	2.7	23.9
QWEN-ROBOTMANIP (Yuan et al., 2026a)	68.6	20.1	14.9	35.9
RLDX-1 (Kim et al., 2026a)	67.6	27.9	8.5	36.0
Xiaomi-Robotics-1 (Team et al., 2026b)	**80.2**	**57.1**	**32.1**	**57.4**
WAM				
GigaWorld-Policy (Ye et al., 2026a)	44.4	11.8	2.9	20.7
ABot-M0.5 (Chen et al., 2026a)	75.9	38.3	2.7	40.4
OpenWAM-α	69.7	32.1	8.9	38.2

Table 19: **Evaluation Results on RoboCasa-GR1.** Bold denotes best values, underline second best.

	SR (%)
VLA	
π_0 (Black et al., 2024)	13.6
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	37.0
GR00T-N1.5 (NVIDIA et al., 2025)	48.0
StarVLA (Community, 2026)	48.8
GR00T-N1.6 (NVIDIA et al., 2025)	49.9
VP-VLA (Wang et al., 2026c)	53.8
Being-H0.5 (Luo et al., 2026a)	53.9
Qwen-VLA-Instruct (Wang et al., 2026a)	56.7
RLDX-1 (Kim et al., 2026a)	58.7
PhysBrain 1.0 (Lin et al., 2026b)	**64.5**
WAM	
UWM (Zhu et al., 2025)	20.0
Being-H0.7 (Luo et al., 2026b)	49.2
DiT4DiT (Ma et al., 2026)	50.8
LDA-1B (Lyu et al., 2026)	55.4
OpenWAM-α	<u>60.5</u>