ONE FROM INFINITY: ACTUALIZING FUTURES FROM PRETRAINED WORLD MODELS INTO ROBOT ACTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

A pretrained video world model admits many plausible futures for a scene, but a robot must realize the exact task-conditioned one. To turn world models into executable robot policies, existing methods fine-tune the heavy world model backbone using large-scale robot data and computational resources. Challenging this *status quo*, we argue that the expensive part has already been paid in the world model pretraining since the representation space of a video world model lays out the diverse potential futures. In this case, what remains is to select the future that accomplishes the task and to read out the actions that realize it. We formalize this task as *actualization*, which learns a task-conditioned selection and realization on top of a prior supplied by a frozen world model. This can be solved by a tiny actualizer model. We implement *RoboActualizer* with as few as 60M parameters on top of a frozen world model encoder. The actualizer is composed of two lightweight DiT experts that jointly predict future latents and actions by flow matching. The model can be trained entirely on a single GPU with 32 GB peak memory. With up to $100\times$ fewer trainable parameters than existing WAMs and VLAs, *RoboActualizer* reaches great performance on simulation benchmarks including LIBERO, LIBERO-Plus, RoboTwin 2.0 and five tasks on two real-world platforms, with a low latency of 39 ms that allows real-time control.



Figure 1: *RoboActualizer* selects the task-conditioned future from the multiple plausible ones contained by the representation space (left). With as few as 60 M parameters, *RoboActualizer* outperforms baselines with $100\times$ more parameters on simulation benchmarks (right).

1 INTRODUCTION

There is an infinity of possible worlds, and only one can exist.

— G. W. Leibniz, *Monadology*

Video world models pretrained on Internet-scale data have become strong predictors of how the world unfolds. From a few context frames, these models (Wan et al., 2025; Assran et al., 2025; Team et al., 2026) learn dense spatio-temporal representations that anticipate future motion, interactions, and scene changes. A defining property of such models is that their predictions are not committed to a single outcome: the same environment admits many possible behaviors, and a general world model must represent all of these potential evolutions rather than a single one.

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.

This property poses a dilemma for robotics, where a robot must ultimately complete one specific task with precise actions. Existing methods bridge this gap at considerable expense. *Planning-based* approaches keep the world model frozen and search over candidate actions at test time, typically with model-predictive control (Zhou et al., 2024; Maes et al., 2026). Each decision then requires many imagined rollouts, which caps the control frequency, and the objective is usually a goal image or a hand-designed cost. Alternatively, *world-action models* (WAMs) fine-tune a general foundation model so that action prediction is absorbed into the world model itself. This, however, requires updating a heavy backbone with substantial robot data and considerable computation (Bjorck et al., 2025; Li et al., 2026; Ye et al., 2026; Yan et al., 2026). Either way, the conversion is costly. Given that enormous computation has already been invested in pretraining these visual foundation models, it is desired that the cost of converting *future predictions* into *robot actions* can be reduced.

In this paper, we seek *an economical way to actualize diverse future predictions into robot actions*. Our key insight is that a pretrained world model encodes the dynamics of the scene and the space of futures it admits, so what remains for the robot is not to learn these dynamics again, but to *select* the one that accomplishes the task among those futures and to read out the actions that realize it (Fig. 1). Selection is a far cheaper problem than dynamics modeling, and we find that *a lightweight module* suffices for it. The world model itself is never updated. All robot-related learning is confined to a tiny flow matching head. Generic pretraining remains an upstream investment, but it is paid once and amortized across tasks, whereas the downstream cost of obtaining a robot policy is reduced to training a tiny model on cached features.

Concretely, we propose *RoboActualizer*, which pairs a frozen V-JEPA 2.1 video encoder (Assran et al., 2025) with two compact diffusion transformers (Peebles & Xie, 2023) organized as a Mixture-of-Transformers (MoT) architecture (Liang et al., 2024). The *latent expert* predicts the future representation of the scene conditioned on the task instruction where the instruction thus acts as the selector to single out from the many futures the world model and admit the one in which the task is accomplished. The future latent provides training-time supervision that teaches the model how to read the frozen representation. The *action expert* predicts, from the same context, the continuous control signals that realize this future. Both are trained jointly from task demonstrations. At deployment, the encoder runs a single forward pass per observation, and only the lightweight head runs repeatedly during denoising. This design gives *RoboActualizer* several key advantages:

- **Strong Performance.** *RoboActualizer* achieves great performance on competitive simulation benchmarks, including LIBERO (Fei et al., 2025b) and RoboTwin 2.0 (Chen et al., 2025), as well as five tasks on two different real-world robots. It consistently outperforms strong baselines despite only 60M trainable parameters (up to 100$\times$ fewer than baselines).
- **Generalization.** On LIBERO-Plus (Fei et al., 2025a), a perturbed variant of LIBERO, *RoboActualizer* reaches 63.1%, compared with 51.5% for Fast-WAM (Yuan et al., 2026) (6.02B trainable parameters). Exploiting the strong representations of the pretrained world model appears to preserve robustness that fine-tuning tends to erode.
- **Low Cost.** *RoboActualizer* has approximately 60M trainable parameters and is optimized on cached latents together, these allowing the entire model to be trained on *a single GPU* with 32 GB peak memory without any embodied pretraining. This substantially lowers the barrier to training world-action models.
- **Efficiency.** Because denoising occurs only within the small actualizer, the deployed policy runs at more than *25 Hz* on a NVIDIA RTX A6000 GPU, satisfying the requirements of real-time robot control.

2 Related Work

Video World Models. Generalizable world models are pretrained on Internet-scale video along two lines. The first generates future frames in pixel space, scaling video generation models into general-purpose simulators (OpenAI, 2024; Agarwal et al., 2025; Wan et al., 2025; Team et al., 2026). The second predicts the future in representation space. The V-JEPA family (Bardes et al., 2024; Assran et al., 2025) learns representations by predicting masked spatio-temporal latents. DINO-WM (Zhou et al., 2024) and NWM (Bar et al., 2025) model dynamics on top of frozen visual features (Oquab et al., 2023). Because latent prediction sidesteps the cost of synthesizing pixels, it yields high-level

semantic features well suited to downstream robotic tasks. Both lines share the property we build on: a world model trained across many environments is not committed to a single continuation, and its predictions span the space of plausible futures. We take a pretrained latent world model as a fixed representation of this space and ask how cheaply a robot policy can be extracted from it.

From World Models to Robot Actions. The predictive ability of world models is an appealing property for robot policies, and existing efforts extract actions from pretrained world models in two ways. *Model predictive control (MPC)* keeps the world model fixed. For each imagined rollout, these methods generate future videos (Du et al., 2023) or latent (Zhou et al., 2024; Maes et al., 2026) and optimize actions against a task-specific reward. Planning preserves the pretrained model but pays the price at inference. Each action requires many rollouts, and the objective is typically a goal image or a hand-designed cost rather than a language instruction. The other approach *fine-tunes* pretrained visual world models into world-action models (WAMs), additionally supervising the policy with future-frame generation in either cascaded or unified form (Cheang et al., 2024; Hu et al., 2024; Bjorck et al., 2025; Ye et al., 2026; Yan et al., 2026). A rapidly growing line keeps future prediction as supervision but moves it from pixels into representation space. FLARE (Zheng et al., 2025) aligns policy features with future latent embeddings. VLA-JEPA (Sun et al., 2026) supervises a VLA with a separate latent world model. Concurrent with our work, ST-WAM (Wang et al., 2026a) adds DINOv3-space (Siméoni et al., 2025) future experts to a video-generative WAM. JEPA-WAM (Lin et al., 2026) couples latent transition prediction and action generation through a shared language-model predictor on top of a frozen encoder. Despite building on V-JEPA 2.1 (Mur-Labadia et al., 2026), JEPA-WAM encodes each observation as a single image and does not condition on an observation history. These works are motivated differently from ours. They treat future image latent as a better supervision target than pixels, but the observation itself is still encoded frame by frame, so the features carry no cues about how the scene is evolving, and all temporal reasoning must be learned by a heavy trainable model. In contrast, *RoboActualizer* encodes a short observation history, so that motion and plausible continuations are already present in the features. What remains is to actualize one of these futures for the task, an easier goal that a tiny trained module suffices for.

Pretrained Visual Representations for Robot Control. A parallel line of work applies general visual representations pretrained on large-scale datasets to robot control without task-specific fine-tuning (Parisi et al., 2022; Goodwin et al., 2022; Xia et al., 2025; Xie et al., 2026), with encouraging results. Others go a step further and learn robot-oriented representations tailored to downstream tasks (Nair et al., 2022; Shang et al., 2024; Wang et al., 2026b). However, frozen pretrained features are sensitive to domain mismatch. Our results suggest a different interpretation: the gap reflects how the representation is read out rather than what it contains. V-JEPA 2.1 (Mur-Labadia et al., 2026) provides dense features that encode diverse possible scene evolutions, and *RoboActualizer* uses them both as policy inputs and as future-prediction targets to learn a lightweight actualizer that selects the task-conditioned future. In this sense, *RoboActualizer* shows that freezing is compatible with strong control when the readout is designed accordingly.

3 METHODOLOGY

This section explains the design of *RoboActualizer* (Fig. 2). We formalize actualization as selecting one future from the many a pretrained world model admits and realizing it in actions (Sec. 3.1), then instantiate the world model with a frozen V-JEPA 2.1 encoder (Sec. 3.2). Sec. 3.3 argues that a narrow actualizer is enough, and Sec. 3.4 presents the detailed implementation of actualizer model.

3.1 FORMULATION: ACTUALIZING ONE FUTURE FROM MANY

Our method rests on a division of labor between a pretrained world model and a small learned actualizer. The world model lays out the plausible futures of a scene, while the instruction picks one of them and produces the actions that realize it.

What the world model provides. The world model provides a representation of the scene that already contains its future dynamcis and evolutions. A video world model g_ϕ, pretrained on large-scale generic video, encodes a short clip of RGB observations into a latent representation

$$z_t = g_\phi(\mathbf{x}_{t-\Delta:t}) \in \mathbb{R}^{N\times D}, \tag{1}$$

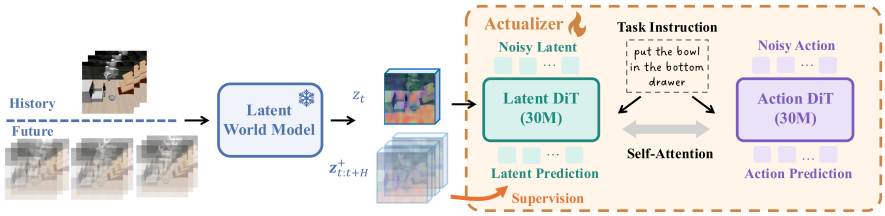

Figure 2: Overview of *RoboActualizer* framework. *RoboActualizer* is built upon a frozen world model. It actualizes the task-conditioned latent future with a lightweight model.

where the clip $\mathbf{x}_{t-\Delta:t}$ spans Δ control steps ending at step t, and z_t consists of N spatial tokens of dimension D. Because g_ϕ attends across frames, z_t encodes the clip as a whole rather than frame by frame. This representation already captures the motion in the window and indicates what can plausibly happen next. The dynamics of the scene are thus built into the representation space, so a downstream learner does not have to acquire them from scratch by itself.

What the world model leaves undetermined. The world model does not determine which of these futures will occur. Knowing what *can* happen is not knowing what *will* happen since the same initial scene can be paired with multiple different tasks (Fig. 5). Let $\mathcal{L}$ be the set of task instructions and $\mathbf{z}^+_{t:t+H}$ be the latent futures over the next H steps. The futures that may follow z_t form a mixture distribution with one component corresponding to each task:

$$p(\mathbf{z}^+_{t:t+H} \mid z_t) \;=\; \sum_{\ell \in \mathcal{L}} p(\ell \mid z_t)\, p(\mathbf{z}^+_{t:t+H} \mid z_t, \ell). \tag{2}$$

Each component is concentrated on the futures in which task ℓ is accomplished. We call the region it occupies $\mathcal{F}_\ell(z_t)$, then the union $\mathcal{F}(z_t) = \bigcup_\ell \mathcal{F}_\ell(z_t)$ is everything the scene can become. Given the latent representation z_t alone, the world model admits all of $\mathcal{F}(z_t)$ but says nothing about which component will be realized.

What actualizer remains to learn. The actualizer learns the task-conditioned choice among these futures and the actions that carry it out. Conditioned on the instruction ℓ, the joint distribution of future and actions factors into three parts with Bayes' rule:

$$p(\mathbf{z}^+_{t:t+H}, \mathbf{a}_{t:t+H} \mid z_t, \ell) \;=\; \underbrace{p(\mathbf{z}^+_{t:t+H} \mid z_t)}_{\textbf{prior}\text{: given by } g_\phi} \underbrace{\frac{p(\ell \mid \mathbf{z}^+_{t:t+H}, z_t)}{p(\ell \mid z_t)}}_{\textbf{selection}\text{: learned}} \underbrace{p(\mathbf{a}_{t:t+H} \mid \mathbf{z}^+_{t:t+H}, z_t)}_{\textbf{realization}\text{: learned}}. \tag{3}$$

From left to right, Eq. (3) says what can happen in future, which of the futures will happen, and how to realize the future. The prior is the mixture of Eq. (2) provided by the world model. g_ϕ was trained to predict exactly such futures, so the region $\mathcal{F}(z_t)$ and its components are already laid out in the frozen space, and the actualizer inherits their geometry rather than learns it. The selection reweights the prior toward the component of task ℓ, and it is the only thing about the future that must be learned from robot data. The realization maps the selected future to the actions that produce it. We call learning the selection and the realization, with the prior held fixed, as *actualization*.

Given robot demonstrations $\mathcal{D}$ of observations, instructions, and actions, the actualizer q_θ is fit to Eq. (3) by maximum likelihood,

$$\min_\theta \; \mathbb{E}_{\mathcal{D}}\Big[-\log q_\theta\big(\mathbf{z}^+_{t:t+H},\, \mathbf{a}_{t:t+H} \,\big|\, z_t, \ell\big)\Big], \qquad \text{s.t. } \; z_t = g_\phi(\mathbf{x}_{t-\Delta:t}) \text{ with } \phi \text{ fixed}, \tag{4}$$

It is implemented as a lightweight flow-matching model in Sec. 3.4. Although Eq. (4) looks similar with an ordinary policy objective, Eq. (3) shows why it is much easier. The targets $\mathbf{z}^+_{t:t+H}$ are points in a frozen space and the prior is already encoded in $\mathbf{z}_t$, so for each instruction, q_θ only has to find the right component $\mathcal{F}_\ell(z_t)$ instead of building the components themselves.

In contrast, previous works fine-tune to fit the same posterior with ϕ among the optimized variables. The geometry of $\mathcal{F}$ is then no longer given, and the learner must shape the prior and the policy at once, which yields a far harder problem. The complexity of actualization, by contrast, is set by the selection rather than by the dynamics of the scene, which is why a tiny q_θ suffices.

3.2 Futures from a Frozen Video World Model

We instantiate g_ϕ with frozen V-JEPA 2.1 (Mur-Labadia et al., 2026), a video foundation model. As a latent world model, the training objective of V-JEPA is to predict masked spatio-temporal latent from context, whose representation space naturally indicates the future prior, as described in Eq. (2).

The present is encoded as a clip. z_t is computed from a short window of frames $\mathbf{x}_{t-\Delta:t}$ ending at t, where we extract the last frame latent as z_t (Eq. (1)). Thus, scene evolution priors enter the representation through the encoder's cross-frame modeling rather than being inferred downstream. Then, the futures are encoded by the same g_ϕ and in the same manner.

The future targets $\mathbf{z}^+_{t:t+H}$ are discretized at a few time steps spread over the action chunk,

$$\mathbf{z}^+_{t:t+H} = (z^+_{t,h})_{h\in\mathcal{H}}, \qquad z^+_{t,h} = g_\phi(\mathbf{x}_{t+h-\Delta:t+h}). \tag{5}$$

Since g_ϕ is frozen in the downstream policy learning, all the latents can be computed offline. We train the actualizer q_θ entirely on cached latents without a single-time forward pass of the video foundation model, which significantly reduces resource requirements and accelerates training.

3.3 Why a Small Actualizer can Work

Before turning to the architecture, we give a representational justification that a narrow actualizer is enough to estimate the distribution of each component in Eq. (2). For a small actualizer with network width $d_{\text{model}}, d_{\text{model}} \ll D$, it can estimate a geometrical component with a rank of at most d_{model}. As a result, the representational ability of the actualizer is bound by the leading n principal components in the representation space of z_t. Fig. 3 evaluates this bound on the latents of Sec. 3.2 extracted from demonstrations in LIBERO Liu et al. (2023). It shows that $d_{\text{model}} = 384$ (width of DiT-S (Peebles & Xie, 2023)) components retain 93.8% of the variance, so the width itself forces a loss of only 6.2%, and it is stable across benchmark suites (shaded band).

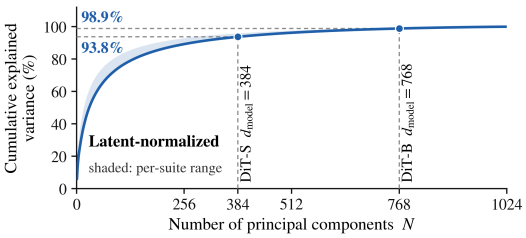

Figure 3: Cumulative explained variance of V-JEPA 2.1 latents. The shaded band is the range across benchmark suites. The dashed line marks the actualizer width $d_{\text{model}} = 384$, at which 93.8% of the variance is retained.

As a result, the V-JEPA representation space is inherently low-dimensional, so a narrow actualizer can nearly losslessly select the task-conditioned components of $\mathcal{F}(z_t)$ and read out actions. However, the naïve *v-prediction* target contains isotropic Gaussian noise of full rank D, which a narrow actualizer cannot capture. We therefore adopt the clean-sample parameterization (a.k.a. *x-prediction*) (Li & He, 2026; Singh et al., 2026), in which the network predicts $\mathbf{z}^+_{t:t+H}$ itself, a target with the same low-rank structure as its input (Appendix A).

3.4 Lightweight Actualizer Model

We implement q_θ as a joint flow-matching model over $(\mathbf{z}^+_{t:t+H}, \mathbf{a}_{t:t+H})$, realized by two lightweight Diffusion Transformers (DiTs) Peebles & Xie (2023) organized as a Mixture-of-Transformers (MoT) architecture (Liang et al., 2024). A *latent expert* predicts the task-conditioned future in latent space along with an *action expert* generates the action chunk that realizes the future. The two experts thus implement the selection and realization factors of Eq. (3) separately, while the prior is carried by the frozen latent space itself. The experts have separate weights and interact through shared self-attention layers. Observation tokens z_t alongside the noisy future latent tokens enter the network through the latent expert, while the noisy action tokens are fed into the action expert. The task instruction is encoded by a frozen UMT5 text encoder (Chung et al., 2023), which together with the encoded proprioceptive state s_t, conditions both experts via cross-attention layers.

As explained at the end of Sec. 3.3, each expert predicts clean tokens as the *x-prediction* target. Following the clean-sample parameterization of Li & He (2026), the latent expert and action expert

Table 1: Success rates (%) on LIBERO benchmark. "Embodied PT" denotes additional embodied data pretraining before downstream task training.

Method	Trainable Params.	Emb. PT	Spatial	Object	Goal	Long	Overall	Latency (ms)
OpenVLA (Kim et al., 2024)	279M	Yes	84.7	88.4	79.2	53.7	76.5	145.5
π_0 (Black et al., 2024)	3.3B	Yes	96.8	98.8	95.8	85.2	94.1	120.4
$\pi_{0.5}$ (Zhou et al., 2025)	3.3B	Yes	**98.8**	98.2	98.0	92.4	96.9	128.5
UniVLA (Bu et al., 2025)	123M	Yes	96.5	96.8	95.6	92.0	95.2	157.3
Motus (Bi et al., 2026)	5.9B	Yes	96.8	99.8	96.6	**97.6**	97.7	2230.0
WorldVLA (Cen et al., 2025)	7.0B	No	87.6	96.2	83.4	60.0	81.8	397.5
Fast-WAM (Yuan et al., 2026)	6.0B	No	98.2	**100.0**	**97.0**	95.2	97.6	111.7
RoboActualizer (ours)	**60M**	No	98.4	**100.0**	96.6	96.8	**98.0**	**39.9**

predict $\hat{\mathbf{z}}^+_{t:t+H}$ and $\hat{\mathbf{a}}_{t:t+H}$ separately, but the training loss is computed in the velocity space:

$$\mathcal{L} = \mathcal{L}_a + \lambda_z \mathcal{L}_z, \qquad \mathcal{L}_a = \mathbb{E}\big\|(\tilde{\mathbf{a}}_\tau - \hat{\mathbf{a}})/(1-\tau_a)\big\|^2, \quad L_z = \mathbb{E}\big\|(\tilde{\mathbf{z}}^+ - \hat{\mathbf{z}}^+)/(1-\tau_z)\big\|^2 \tag{6}$$

where the noisy tokens $\tilde{\mathbf{a}}_\tau = (1-\tau_a)\epsilon_a + \tau_a \mathbf{a}$ and $\tilde{\mathbf{z}}^+_\tau = (1-\tau_z)\epsilon_z + \tau_z \mathbf{z}^+$ with gaussian noise $\epsilon_a, \epsilon_z \sim \mathcal{N}(0, I)$ and noise level $\tau_a, \tau_z \in (0, 1)$. λ_z is a hyper-parameter to balance the two losses.

At each control step, the encoder is passed once to obtain z_t, and only the actualizer is iterated over K denoising steps to produce the action chunk. With denoising confined to the small head, the policy runs at over 25 Hz on a single NVIDIA A6000 GPU.

4 Experiments

We conduct extensive experiments to evaluate the performance of *RoboActualizer* in both simulation and real environments. Sec. 4.1 describes the benchmarks, baselines, and protocol. Sec. 4.2 provides the simulation results including three benchmarks: LIBERO, LIBERO-Plus, and RoboTwin 2.0. Sec. 4.3 deploys *RoboActualizer* on two physical platforms, a parallel-gripper arm and a bimanual system with dexterous hands. Sec. 4.4 quantifies the training cost and inference latency. Finally, Sec. 4.5 gives a deep dive into how *RoboActualizer* works and provides ablation studies.

4.1 Experimental Setup

Model Details. Unless otherwise stated, we adopt a pretrained V-JEPA 2.1 (Mur-Labadia et al., 2026) (ViT-L/16, 300M) as the video world model, while the actualizer is implemented with a DiT-S model (60M) (Peebles & Xie, 2023). The actualizer is randomly initialized for each benchmark without any additional pretraining. More details are in the appendix.

Benchmarks. We evaluate *RoboActualizer* on three simulation suites and two real-world platforms. The simulation benchmarks include: (i) **LIBERO** (Liu et al., 2023): four task suites (Spatial, Object, Goal, Long) for robot manipulation; (ii) **LIBERO-Plus** (Fei et al., 2025a): a perturbed variant of LIBERO to evaluate the generalization ability in out-of-distribution situations; (iii) RoboTwin 2.0 (Chen et al., 2025): a large-scale bimanual manipulation benchmark. We also conduct experiments on two different real-world robot hardware: (i) **FANUC CRX-10iA arm**: we conduct two single-arm manipulation tasks; (ii) **Unitree G1**: we apply three bimanual dexterous manipulation tasks following prior work (Chen et al., 2026).

Baselines. On simulation benchmarks, we compare against multiple previous works in the field of vision-language-action (VLA) model and world-action model (WAM). For real-world robots, we implement two representative robot policy baselines: $\pi_{0.5}$ (Zhou et al., 2025) as the VLA baseline, and Fast-WAM (Yuan et al., 2026) as the WAM baseline.

4.2 Generalization in Simulation

LIBERO Benchmark. In Table 1, we compare the performance of *RoboActualizer* with baselines on LIBERO benchmark (Liu et al., 2023). We train a single *RoboActualizer* model on the four standard evaluation suites (10 tasks per suite). With only 60M trainable parameters, *RoboActualizer* achieves great performance with a 98.0% average success rate. This performance outperforms multiple existing VLA or WAM baselines that have $100\times$ more trainable parameters.

Table 2: Success rates (%) on LIBERO-Plus benchmark. *: no open-source code to measure latency.

Method	Trainable Params.	Camera	Robot	Lang.	Light	Bg.	Noise	Layout	Overall	Latency (ms)
OpenVLA (Kim et al., 2024)	279M	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6	145.5
π_0 Black et al. (2024)	3.3B	13.8	6.0	58.8	85.0	**81.4**	**79.0**	68.9	53.6	120.4
π_0-Fast	3.3B	**65.1**	21.6	61.0	73.2	73.2	74.4	68.8	61.6	68.5
UniVLA (Bu et al., 2025)	123M	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9	157.3
WorldVLA (Cen et al., 2025)	7.04B	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0	397.5
Fast-WAM (Yuan et al., 2026)	6.02B	16.4	44.5	68.9	78.2	53.7	37.7	60.7	51.5	111.7
DC-WAM (Ji et al., 2026)	6.00B	23.9	**51.7**	**83.4**	**91.7**	61.3	54.2	**69.8**	60.9	–*
RoboActualizer (ours)	**60M**	37.6	51.1	68.6	85.1	60.4	54.5	60.9	**63.1**	**39.9**

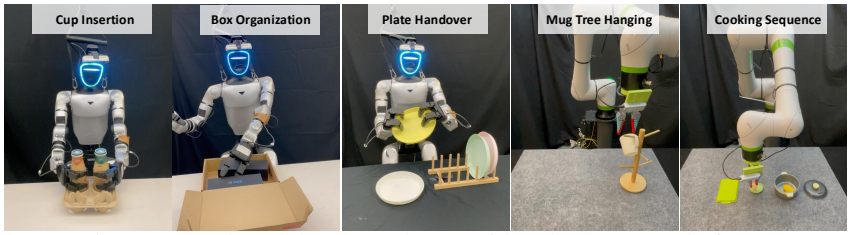

Figure 4: Real-world tasks on humanoid and robot arm platforms.

LIBERO-Plus Benchmark. LIBERO-Plus (Fei et al., 2025a) is a more challenging benchmark built up on LIBERO tasks with notable visual and layout variations. To evaluate *RoboActualizer* on out-of-distribution situations, we only train *RoboActualizer* on LIBERO training demonstrations without any additional data augmentations that fit the shifted distribution. Results in Table 2 reflect the strong generalization ability of *RoboActualizer* despite its small trainable parameter count. We suppose *RoboActualizer* can exploit the strong generalization ability of the pretrained video world model, and only actualize the task-conditioned one with its limited network capacity.

RoboTwin 2.0 Benchmark (50 demos per task). We further evaluate *RoboActualizer* under the RobotWin 2.0 benchmark (Chen et al., 2025), in which tasks are much more challenging than LIBERO (Fei et al., 2025b). Limited by computational resources, we randomly sample 50 demonstrations across the 50 tasks. This setting could also evaluate the generalization ability of *RoboActualizer* with limited training data. As shown in Table 3, *RoboActualizer* greatly outperforms the baselines. Thanks to its small size, *RoboActualizer* exhibits better data efficiency and avoids the overfitting issue.

Table 3: Success rates (%) on RoboTwin 2.0 (50 demos per task).

Method	Trainable Params.	Success Rates (%)
$\pi_{0.5}$ (Zhou et al., 2025)	3.3B	31.4
LingBot-VA (Li et al., 2026)	5.3B	17.2
Fast-WAM (Yuan et al., 2026)	6.0B	41.9
RoboActualizer (ours)	60M	**58.8**

4.3 Real-World Deployment

As shown in Fig. 4, we evaluated our method on five real-world tasks across two platforms: a G1 humanoid with a Unitree Dex3-1 hand for bimanual manipulation and a FANUC CRX-i10 arm with a gripper for single-arm manipulation. The G1 operated with its lower body stationary and used a head-mounted RealSense camera. The FANUC used two third-person cameras and one in-hand camera. We collected approximately 170 demonstrations per humanoid task and 100 per single-arm task, reflecting their differences in difficulty and action dimensionality (details in Appendix. D). Fast-WAM and *RoboActualizer* are trained from scratch, while $\pi_{0.5}$ is fine-tuned from its pretrained weight. Table 4 shows that our method consistently outperforms the corresponding baselines across all real-world tasks, demonstrating effective learning across single-arm and bimanual platforms. Videos are in the supplementary materials.

4.4 Cost and Efficiency

Single-GPU Model Training. We train *RoboActualizer* on robot data on the top of cached V-JEPA latent features which relieves the burden of repetitive world model inference. We only optimize as few as 60M model parameters of the actualizer, which allows us to finish the training on a single NVIDIA PRO 6000 GPU with 32 GB peak memory in only 15 hours on LIBERO Benchmark (Liu et al., 2023). In contrast, our WAM baseline, Fast-WAM (Yuan et al., 2026), has 6B parameters and requires an estimated 112 GPU-hours under its reported LIBERO training configuration.

Table 4: Success rates (%) in real-world experiments.

(a) FANUC CRX-10iA (20 rollouts)

Method	Mug	Cooking
Fast-WAM	10	15
RoboActualizer	**95**	**60**

(b) Unitree G1 (30 rollouts)

Method	Box	Cup	Plate
$\pi_{0.5}$	57	50	30
RoboActualizer	**77**	**83**	**90**

Table 5: Ablation on latent-action joint prediction

Latent Prediction	Success Rates (%)
✗	17.9
✓	**60.0**

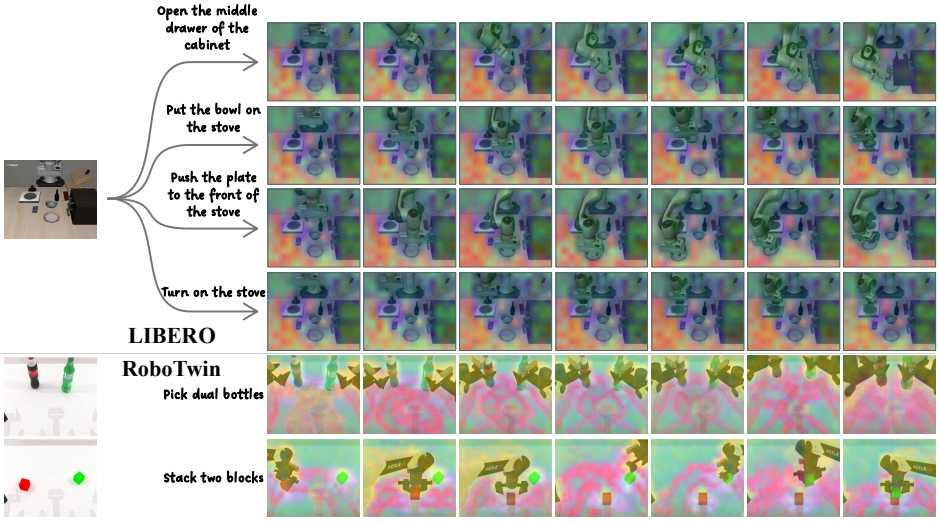

Figure 5: Predicted versus ground-truth future latent prediction at multiple time steps, visualized with PCA dimension reduction. Features predicted by latent expert tracks the teacher outputs greatly.

Efficient Inference. In the deployment, the V-JEPA encoder only runs inference once, while the lightweight actualizer executes the repetitive denoising steps. Inspired by Fast-WAM (Yuan et al., 2026), the future latent prediction is discarded, so the actualizer only predicts the action and intermediate features of the clean frame can be cached in the denoising process. In Table 1 and Table 2, we measure the latency of all the methods on a single NVIDIA RTX A6000 GPU. *RoboActualizer* yields a latency of 39 ms, significantly faster than all the previous works. This enables *RoboActualizer* to easily satisfy the requirements of real-time robot control. In Fig. 6, most of *RoboActualizer*'s latency is occupied by the one-time video encoder, and the denoising latency is very small.

4.5 What Enables Lightweight Actualization?

We study why a small actualizer can effectively read out a frozen pretrained representation space, including qualitative and quantitative analysis. All the ablation studies are measured with success rates on LIBERO-Plus (Fei et al., 2025a). Due to some slightly different setups, the success rates are different with Table. 2, but the comparison fairness is guaranteed.

4.5.1 How Lightweight Actualizer Helps?

Our proposed method actualizes the task-conditioned future and produces the actions to realize it. In Table 5, removing the latent prediction target significantly reduces the success rate from 60% to 17.9%. This action-only prediction is equivalent to attaching a flow matching policy on top of the same frozen V-JEPA representation, which notably hurts the generalization ability. This reflects that the frozen V-JEPA representation itself is not strong enough for the robot policy but our joint prediction design fully stimulates the inherent future predictive ability from pretrained world models. Visualizations for future latent prediction are shown in Fig. 5. The features are projected onto their first three PCA components for RGB visualization. *RoboActualizer* could predict both the task-conditioned future latent and diverse futures conditioned on different task instructions faithfully.

4.5.2 How *RoboActualizer* Different from Latent World Action Models?

Several previous or concurrent works (Wu et al., 2026; Yan et al., 2026; Lin et al., 2026) have explored joint prediction of future latents and actions. However, *RoboActualizer* comes from a

Table 6: Ablation study on latent representation. The ability for future prediction inside the latent space is more important than its representation space.

Representation	Params.	Input Range	Latent Predictivity	Success Rates (%)
DINOv3	300M	1 frame	✗	39.7
WAN VAE	127M	4 frames	✗	25.9
V-JEPA 2.1_{image}	300M	1 frame	✗	46.4
V-JEPA 2.1_{image}	300M	4 frames	✗	44.8
V-JEPA 2.1_{video}	300M	4 frames	✓	**60.0**

Table 7: Computation allocation between world model and actualizer.

World Model	Actualizer	Success Rates (%)
V-JEPA 2.1 (80M)	DiT-S (60M)	50.7
V-JEPA 2.1 (300M)	DiT-S (60M)	60.0
V-JEPA 2.1 (300M)	DiT-B (245M)	**61.0**

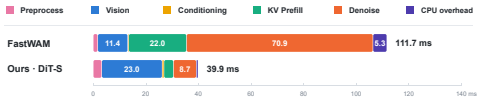

Figure 6: Inference latency split of Fast-WAM and *RoboActualizer*.

fundamentally different motivations. Other works consider latent features as a superior alternative to RGB images due to their rich high-level semantics. As a result, they still require huge models with billions of parameters to learn the future predictions. In contrast, *RoboActualizer* makes full use of the inherent future predictive ability inside the video world model representation space, so *RoboActualizer* only chooses rather than imagines task-conditioned futures. This largely simplifies the learning of our actualizer, and thus enables an extremely lightweight model. In Table 6, we conduct an ablation study to deprive the future prediction ability by changing the latent representation space. Results show that other representation space (e.g. DINO) brings bad performance despite its strong representative ability. Further, we adopt the same V-JEPA 2.1 encoder but apply to single image frame separately, which ensures the same representation space but strips the inherent future indication. This also brings inferior performance in Table 6. We could conclude that the future prediction ability inside the latent is much more important than its representative ability. This also justifies our motivation to actualize the task-conditioned future from the diverse predictions rather than learn the future prediction that relieves the computation cost for training the actualizer.

4.5.3 How to allocate the computation

As our model is composed of a pretrained video world model and a trainable actualizer, it is useful to discuss how to allocate our computation. Based on the results in Table 7, we find it more beneficial to scale up the pretrained world model rather than the trainable actualizer, although the world model is always frozen. This fact suggests that the future prediction ability of the frozen world model is critical for the robot policy. We do not need a powerful actualizer, as it only needs to select the task-conditioned future among those admitted by the world model rather than learning to predict futures by itself. Additional ablations of the architectural design are reported in Appendix A.

5 Conclusion

We have presented *RoboActualizer*, built on a division of labor: a frozen video world model lays out the plausible futures of a scene, and a small actualizer selects the task-conditioned one and realizes it as actions. Under this design, priors over future dynamics are inherited from the frozen representation space, and what must be learned from robot data reduces to a task-conditioned selection and a readout of actions, which yields a problem of far lower complexity than predicting scene evolution itself. We instantiate the actualizer as a compact DiT (60M trainable parameters) with an MoT architecture that jointly predicts latent features and actions conditioned on task instructions. Our experiments show that this small model outperforms VLA and WAM baselines with billions of trainable parameters on LIBERO, LIBERO-Plus, RoboTwin 2.0, and two real-world setups.

Limitations. Due to limited resources, we do not conduct experiments on the complete RoboTwin 2.0 benchmark (Chen et al., 2025). Besides, we only apply our method to V-JEPA 2.1 video world model. We will list the combination of *RoboActualizer* with other world model and experiments on large-scale benchmarks in our future works.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools for research ideation or the development of the core research ideas of this work, and generating synthetic datasets, formulating mathematical claims, and proving mathematical claims are not applicable to this work. Additionally, we used generative AI tools for creating and editing software code, creating research artifacts including tables and supplementary materials, drafting portions of the manuscript, and editing the paper for readability. We have reviewed all AI-assisted work. We verified the correctness of AI-assisted outputs and their consistency with our experiments and analyses. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The main paper describes the observation construction, prediction targets, training objective, and hardware requirements. Appendix B summarizes the confirmed implementation settings. We will release the code when the paper is accepted.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801. IEEE, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 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.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025. URL https://arxiv.org/abs/2505.06111.

Jun Cen, Chaohui Yu, Hangjie Yuan, et al. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025. URL https://arxiv.org/abs/2506.21539.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

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.

Yuxin Chen, Hari Srikanth, Nathan Jew, Menglin Wu, Pengcheng Wang, Junli Ren, Masayoshi Tomizuka, Peng Xu, Jinyu Xie, and Thomas Tian. Clift: Turning gemini robotics on-device into humanoid specialists via non-invasive closed-loop iterative fine-tuning. *arXiv preprint arXiv:2607.29172*, 2026.

Hyung Won Chung, Xavier Garcia, Adam Roberts, Yi Tay, Orhan Firat, Sharan Narang, and Noah Constant. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=kXwdL1cWOAi.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

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*, 2025a. URL https://arxiv.org/abs/2510.13626.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025b.

Walter Goodwin, Sagar Vaze, Ioannis Havoutis, and Ingmar Posner. Zero-shot category-level object pose estimation. In *European Conference on Computer Vision*, pp. 516–532. Springer, 2022.

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

Haoyuan Ji, Lingxiang Fan, Shang Su, Yinqiao Lu, Mengkai Shi, Jun Gao, and Shuo Feng. Dcwam: Dynamic-centric visual supervision and reasoning for world-action models. *arXiv preprint arXiv:2607.25918*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

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

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Yihan Lin, Jiawei He, Shifeng Bao, Chen Zhao, Yang Li, Xiaobo Wang, Yan Wang, Cheng Chi, and Jing Zhang. JEPA-WAM: Learning vision-language-action policies with joint-embedding world modeling. *arXiv preprint arXiv:2608.09381*, 2026. URL https://arxiv.org/abs/2608.09381.

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. URL https://arxiv.org/abs/2306.03310.

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.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike 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. URL https://arxiv.org/abs/2603.14482.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. URL https://arxiv.org/abs/2203.12601.

OpenAI. Video generation models as world simulators. https://openai.com/index/video-generation-models-as-world-simulators/, 2024. Accessed: 2026-09-22.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *international conference on machine learning*, pp. 17359–17371. PMLR, 2022.

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.

Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. *arXiv preprint arXiv:2407.20179*, 2024.

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

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

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, et al. Advancing open-source world models. *arXiv preprint arXiv:2601.20540*, 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.

Mingxin Wang, Bin Hu, Bin Qian, Kaitao Jiang, Haoning Wu, Feng Yan, Bowen Jing, Ruiyang Hao, Enyi Wang, Kangning Niu, Yandan Yang, Mu Xu, Yan Wang, Houde Liu, and Tianlun Li. ST-WAM: Semantic-temporal world action model for robust manipulation under visual distribution shifts. *arXiv preprint arXiv:2607.28993*, 2026a. URL https://arxiv.org/abs/2607.28993.

Yixiao Wang, Mingxiao Huo, Zhixuan Liang, Yushi Du, Lingfeng Sun, Haotian Lin, Jinghuan Shang, Chensheng Peng, Mohit Bansal, Mingyu Ding, et al. Ver: Vision expert transformer for robot learning via foundation distillation and dynamic routing. In *International Conference on Learning Representations*, volume 2026, pp. 155122–155148, 2026b.

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

Shangning Xia, Hongjie Fang, Cewu Lu, and Hao-Shu Fang. Cage: Causal attention enables data-efficient generalizable robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13242–13249. IEEE, 2025.

Yichen Xie, Yixiao Wang, Shuqi Zhao, Cheng-En Wu, Masayoshi Tomizuka, Jianwen Xie, and Hao-Shu Fang. Multi-camera view scaling for data-efficient robot imitation learning. *arXiv preprint arXiv:2604.00557*, 2026.

Ge Yan, Jinghao Liu, Yuzhi Fan, Lei Cai, Minwen Liao, Jesse Zhang, and Dieter Fox. Flex-π: A multi-stream world-action model with compute flexibility, 2026. URL `https://arxiv.org/abs/2608.10860`.

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

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL `https://arxiv.org/abs/2603.16666`.

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, 2026.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

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.

Zhongyi Zhou, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Vision-language-action model with open-world embodied reasoning from pretrained knowledge. *arXiv e-prints*, pp. arXiv–2505, 2025.

A Additional Architecture Ablations

Mixture-of-Transformers Architecture. We ablate the MoT architecture. The single-DiT variant shares weights across the latent and action experts, while keeping the other design choices unchanged. Table 8a shows that this substantially reduces performance: the LIBERO-Plus success rate drops from 60.0% to 51.8%. We hypothesize that the MoT architecture better separates the selection and realization processes in Eq. 3.

Prediction Target. Table 8b shows that the *x-prediction* target is important for strong performance. This is consistent with our analysis in Sec. 3.3: *x-prediction* allows the latent space to be represented with a narrow network.

Wide Head. Given the high-dimensional latent space of V-JEPA 2.1, one possible approach is to use a wide head, as in RAE (Zheng et al., 2026; Singh et al., 2026). However, this design does not improve *RoboActualizer* (Table 8c). As discussed in Sec. 3.3, the small actualizer is sufficient for actualization, and a wide head may make the training process unstable.

Table 8: Ablation study on model architecture design.

(a) Mixture of Transformers

Method	Success Rate (%)
Single DiT	51.8
MoT	**60.0**

(b) Prediction target

Target	Success Rate (%)
v-prediction	52.7
x-prediction	**60.0**

(c) Wide head

Variant	Success Rate (%)
w/ wide head	48.5
w/o wide head	**60.0**

B Implementation and Reproducibility Details

B.1 Model Architecture Details

Frozen visual encoder. All experiments use a frozen V-JEPA 2.1 ViT-L/16-384 encoder ($\sim$300M parameters, distilled from ViT-g), kept in `float32` and in `eval` mode at all times; it is excluded from the optimizer and from the released checkpoints. Inputs are mapped from the $[-1, 1]$ convention used throughout our pipeline to ImageNet normalization inside the backbone wrapper. A frame is encoded by the *video* branch (tubelet 2) rather than the single-image branch; the clip construction is described in Sec. B.2.

Denoiser. The denoiser is a two-branch mixture-of-transformers: an *image branch* and an *action branch*, each with 12 pre-norm blocks. At every layer the two branches compute their own q, k, v and are then mixed by a *single joint self-attention* over the concatenated token stream, after which each branch independently applies cross-attention to its own conditioning tokens and its own FFN. Blocks use AdaLN-single conditioning (Wan style): one zero-initialized $\mathrm{SiLU} \to \mathrm{Linear}(d, 6d)$ head per branch is shared by all 12 layers, and each layer adds a learnable constant $\mathbf{m} \in \mathbb{R}^{6\times d}$, yielding shift/scale/gate for attention and FFN. Attention uses RMSNorm on q and k ($\epsilon = 10^{-6}$), FFNs are $\mathrm{Linear} \to \mathrm{GELU} \to \mathrm{Linear}$ with ratio 4, the pre-cross-attention LayerNorm is affine while the AdaLN-modulated LayerNorms are affine-free, dropout is 0, and no long skip connections are used. Two sizes follow the official DiT configurations (depth 12, head dim 64):

	d_{model}	heads	FFN	joint expert	trainable params
DiT-S	384	6	1536	56.80M	63,604,487
DiT-B	768	12	3072	226.84M	245,762,567

The remaining parameters are the per-branch context encoders (2×1.87M / 2×4.33M), timestep embedders (2×1.13M / 2×4.33M), the shared latent projection, and the two output heads.

Positional encoding. Positions are encoded exclusively by RoPE (no additive embeddings). With head dimension 64 (32 complex pairs), image tokens use 3D RoPE split $12/10/10$ pairs over (frame, height, width); the frame axis carries the *actual* environment step, $[0, 4, 8, 12, 16]$ for z_0 and the four targets. Action tokens use 1D RoPE at positions $0, \ldots, 15$, so that action step k and image frame j live on the same step axis. Rotations are applied to q, k (never v) after qk-normalization, in `float64`, then cast back.

Conditioning. Language embeddings are precomputed once with the frozen umT5-XXL text encoder of Wan2.2-TI2V-5B: 128 tokens $\times$ 4096 dims, using the prompt template *"A video recorded from a robot's point of view executing the following instruction: {task}"*. Padded positions are zeroed and masked out of cross-attention. Proprioception (8-D) is embedded by $\mathrm{Linear} \to \mathrm{GELU} \to \mathrm{Linear}$ into one token appended to the text tokens, giving 129 context tokens. The two branches own *separate* text and proprio encoders (no weight sharing); a single conditioning stream is shared across all 12 layers of a branch.

Output heads. The image head is $\mathrm{LN}_{\text{no-affine}} \to$ AdaLN modulate $\to \mathrm{Linear}(d, 1024)$, where shift/scale are a learnable $[2, d]$ constant plus the timestep embedding; the action head is a single $\mathrm{Linear}(d, 7)$. Both heads are zero-initialized, so the network predicts $\hat{x} = 0$ at initialization.

Latent normalization. Before entering the denoiser, every V-JEPA latent is normalized per channel, $\tilde{z} = (z - \mu)/\sigma$ with $\mu, \sigma \in \mathbb{R}^{1024}$ computed in `float64` over the entire cache (1.11×10^{11} stored fp16 values; global std 1.84, per-channel $|\mu|$ up to 24). The statistics are registered as buffers and travel with the checkpoint, so training and evaluation provably share one set of constants. The same transform is applied to z_0 at inference.

B.2 Observation and Target Construction

Data. We use the four LIBERO suites (Spatial, Object, Goal, Long) in their no-noops LeRobot conversion at 20 Hz: 1,712 demonstrations and 277,713 frames (mean episode length 122.6/147.3/122.2/268.8 respectively). One training sample is anchored at every frame (stride 1), so the training set contains 277,713 windows.

Image pipeline. Raw 512×512 frames are decoded, converted to tensors and resized per camera to 224×224, concatenated horizontally, then resized/center-cropped to 224×448 and normalized to $[-1, 1]$ — the identical pipeline is used for cache generation, online training and closed-loop evaluation.

Context latent z_0. z_0 is not a single-frame embedding: the clip $\{t-6, t-4, t-2, t\}$ (`clip_len`$=4$, `frame_interval`$=2$) is passed through the V-JEPA *video* branch (tubelet 2), and the last temporal group is kept, giving 392×1024 tokens that carry short-horizon motion while remaining index-aligned with the action/proprio row of frame t. Indices before the episode start are clamped to the first valid frame (i.e. the earliest frame is repeated).

Prediction targets. The window spans 17 frames; with `action_video_freq_ratio`$=4$ the image branch supervises the latents at $t+4, t+8, t+12, t+16$ (`future_size`$=4$), each encoded with the same causal-clip rule as z_0. The action branch supervises the next 16 normalized actions (`chunk_size`$=16$), i.e. the action chunk and the image targets cover the same 16-step horizon.

Action and state spaces. Actions are 7-D: a 6-D delta end-effector pose (OSC) plus an absolute gripper command. Proprioception is 8-D: EE pose (6) plus the two gripper joints, taken at the anchor frame only. Both are normalized per dimension to $[-1, 1]$ with min/max statistics computed over the training split (the normalizer clamps to ± 5; it is exactly invertible for evaluation). For padded steps the six delta dimensions are zeroed before normalization.

B.3 Training Details

Objective. We use a shifted rectified-flow schedule with 1000 discrete levels and $\sigma = \varphi(u) = su/(1 + (s-1)u)$, $s = 5$ for both branches ($\sigma = 1$ noise, $\sigma = 0$ clean). Noising is $z_\sigma = (1-\sigma)x + \sigma\varepsilon$; the network predicts the clean sample $\hat{x}$ and the loss is taken in velocity space, $\hat{v} = (z_\sigma - \hat{x})/\max(\sigma, 0.05)$ against $v = (z_\sigma - x)/\max(\sigma, 0.05)$. Prediction and target share the same clamped denominator, so this is exactly an $1/\sigma^2$-weighted x-loss with a floor at $\sigma = 0.05$. Timesteps are drawn *independently* for the two branches, covering the whole (σ_a, σ_r) square; the network receives $t = \sigma \cdot 1000$ through a 256-dim sinusoidal embedding followed by a two-layer MLP, and the clean prefix z_0 is conditioned with $t = 0$. Both terms are per-token MSE, averaged

over the feature axis, masked-averaged over valid steps/frames, then averaged over the batch, and combined as $\mathcal{L} = 1.0 \cdot \mathcal{L}_{\text{action}} + 1.0 \cdot \mathcal{L}_{\text{image}}$.

Shared optimization protocol. All configurations share: AdamW ($\beta = (0.9, 0.95)$, weight decay 10^{-2}), peak learning rate 2×10^{-4}, linear warm-up over the first 5% of the total steps then cosine decay to 1% of the peak, batch size 32 on a single GPU with no gradient accumulation, bf16 mixed precision , seed 42, no EMA, no dropout, and the same data pipeline (Sec. B.2): 16-step action chunks, four future latents at $t{+}4, \ldots, t{+}16$, per-channel latent normalization, and $277{,}713$ training windows ($8{,}679$ optimizer steps per epoch). Only the visual encoder, the denoiser width, the epoch budget and the gradient-clipping threshold differ across rows. Checkpoints are written every $5{,}000$ steps and we always report the *final* checkpoint; no checkpoint selection on evaluation results is performed.

Encoder	Denoiser	D_z	trainable	epochs	steps	grad. clip
V-JEPA 2.1 ViT-B/16 (80M)	DiT-S	768	63.41M	20	173,580	1.0
V-JEPA 2.1 ViT-L/16 (300M)	DiT-S	1024	63.60M	20	173,580	1.0
V-JEPA 2.1 ViT-L/16 (300M)	DiT-B	1024	245.76M	15	130,185	2.0
V-JEPA 2.1 ViT-G/16 (2B)	DiT-B	1664	246.75M	15	130,185	1.0

C Evaluation Protocols and Baseline Provenance

All evaluations are closed-loop in the original simulators with a single policy implementation. No evaluation-time tuning, action ensembling, or test-time augmentation is used. Common inference settings are listed in Table 9.

Table 9: Closed-loop inference settings.

	LIBERO	LIBERO-Plus	RoboTwin 2.0
Action chunk / executed steps	16 / 10	16 / 10	32 / 24
Euler steps	4	4	10
CFG scale	1	1.5	1
Future-image generation	off	off	off
Warm-up no-op steps	30	30	–

C.1 LIBERO and LIBERO-Plus

Dataset and evaluation. We evaluate on LIBERO and LIBERO-Plus. For LIBERO, we jointly train on the four standard task suites (Spatial, Object, Goal, Long; $1{,}712$ demonstrations, $277{,}713$ frames) and evaluate the resulting policy on the corresponding test tasks: 10 tasks per suite with 50 trials each, i.e. $2{,}000$ episodes per checkpoint. Initial states are taken from the benchmark's own init-state files indexed by trial number, so every model sees identical start configurations, and the simulator is seeded with 42. For LIBERO-Plus, we directly transfer the policy trained on LIBERO demonstrations to the perturbed environments without additional fine-tuning. LIBERO-Plus provides $10{,}030$ perturbed task instances over seven perturbation dimensions (Camera, Robot, Language, Light, Background, Noise, Layout), each evaluated for a single episode; we report the success rate per dimension and their unweighted mean. For ablations we use a frozen stratified subsample of $2{,}640$ instances (660 per suite, balanced over the seven dimensions, drawn once with seed 42 and reused by every run), which reproduces the full-benchmark mean to within 0.3 points.

Observation and action configuration. The policy receives images from the primary camera and the wrist camera together with language instructions and proprioceptive states. The two views are rendered at 256×256, resized to 224×224 each and concatenated horizontally into a 224×448 input, which the frozen V-JEPA encoder maps to 392 visual tokens. Language is encoded offline into 128 tokens and proprioception is 8-dimensional (end-effector pose and two gripper joints). The action space contains 7-dimensional robot actions (6 OSC pose deltas and one gripper command) with an action chunk length of 16.

Closed-loop execution. Each episode begins with 30 no-op steps so the scene settles. The policy then denoises a chunk with 10 Euler steps, executes the first 10 actions, and replans; an observation is taken after every simulator step so that the four-frame, stride-two clip conditioning the policy keeps the temporal spacing used in training. Predicted actions are de-normalized with the checkpoint's own dataset statistics and the gripper channel is binarized. The episode budget is 400 steps for Spatial, Object and Goal and 700 steps for Long. Training hyper-parameters are given in Sec. B.3.

C.2 RoboTwin 2.0

Dataset and evaluation. We evaluate on RoboTwin 2.0 using all 50 manipulation tasks. For each task the policy is trained on a fixed subset of 50 demonstrations (2,500 episodes, 564,911 frames in total, selected once with seed 42). The same trained policy is then evaluated on both the Clean and the Random settings, where Random introduces randomized backgrounds, a cluttered table, randomized lighting and table-height jitter. We run 25 episodes per task and setting (2,500 evaluation episodes) and report the average task success rate over all 50 tasks. Language instructions at evaluation are drawn from the unseen template pool, and evaluation scenes are disjoint from training scenes by construction: episode seeds start at 4,300,000 whereas the demonstrations were collected from seeds starting at 0. Per-episode step limits and success criteria are the benchmark's own.

Observation and action configuration. RoboTwin provides one external camera and two wrist cameras at 240×320. The three views are composed into a single 384×320 canvas (external view on top, the two wrist views side by side below) and encoded by the frozen V-JEPA encoder into 480 visual tokens. The policy predicts 14-dimensional bimanual actions with an action chunk length of 32, of which 24 steps are executed before replanning; denoising uses 10 Euler steps. Training follows Sec. B.3 with a longer temporal stride (clips of four frames at stride four, future latents at $+8, \ldots, +32$) to match the 50 Hz control rate.

Per-task results. Complete task-wise results for all 50 tasks are given in Table 10; the overall success rate is 57.8% (Clean), 59.9% (Random) and 58.8% on average.

Table 10: RoboTwin 2.0 per-task success rate (%), 25 episodes per cell. C = Clean, R = Random, M = mean.

Task	C	R	M	Task	C	R	M
adjust_bottle	100	100	100	place_can_basket	64	64	64
beat_block_hammer	76	64	70	place_cans_plasticbox	56	84	70
blocks_ranking_rgb	36	36	36	place_container_plate	100	96	98
blocks_ranking_size	24	32	28	place_dual_shoes	64	44	54
click_alarmclock	92	84	88	place_empty_cup	72	72	72
click_bell	100	92	96	place_fan	40	56	48
dump_bin_bigbin	88	88	88	place_mouse_pad	28	20	24
grab_roller	80	96	88	place_object_basket	52	60	56
handover_block	64	36	50	place_object_scale	44	12	28
handover_mic	92	96	94	place_object_stand	76	72	74
hanging_mug	28	32	30	place_phone_stand	48	60	54
lift_pot	44	60	52	place_shoe	76	68	72
move_can_pot	48	68	58	press_stapler	60	84	72
move_pillbottle_pad	44	32	38	put_bottles_dustbin	36	52	44
move_playingcard_away	64	68	66	put_object_cabinet	36	64	50
move_stapler_pad	8	8	8	rotate_qrcode	72	68	70
open_laptop	88	88	88	scan_object	24	36	30
open_microwave	20	24	22	shake_bottle	100	100	100
pick_diverse_bottles	36	48	42	shake_bottle_horizontally	100	100	100
pick_dual_bottles	68	52	60	stack_blocks_three	8	28	18
place_a2b_left	36	36	36	stack_blocks_two	80	84	82
place_a2b_right	56	40	48	stack_bowls_three	64	60	62
place_bread_basket	36	44	40	stack_bowls_two	100	92	96
place_bread_skillet	60	52	56	stamp_seal	16	28	22
place_burger_fries	56	84	70	turn_switch	28	32	30
Overall (50 tasks)					**57.8**	**59.9**	**58.8**

D Real-Robot Experiments

We evaluate *RoboActualizer* on two real-world platforms: a Unitree G1 humanoid for bimanual manipulation and a Fanuc CRX-10iA arm for tabletop manipulation. Unless otherwise specified, all

experiments use the 300M V-JEPA encoder, DiT-S (63.6M trainable parameters), single-frame conditioning, and four future latent targets at $+4, +8, +12, +16$ steps. Optimization follows Sec. B.3.

D.1 UNITREE G1

Tasks and data. We consider three bimanual manipulation tasks on a Unitree G1 humanoid: *Bimanual Plate Handover*, *Box Packing*, and *Cup Insertion*. Demonstrations are successful policy rollouts recorded at 30,Hz and converted to LeRobot v2.1 format. We train a separate policy for each task.

Observations and actions. The policy receives synchronized RGB observations from a third-person camera and a head-mounted egocentric camera. Both 640×480 images are center-cropped, resized to 224×224, and concatenated horizontally into a 224×448 input, yielding 392 V-JEPA tokens. Proprioception and actions are 16-dimensional absolute joint-position vectors, comprising 7 joints per arm and two binary gripper states. Both are min–max normalized. The policy predicts 16 actions per chunk, corresponding to 0.53,s at 30,Hz.

Task	Episodes	Frames	LR	Epochs	Steps
Plate Handover	167	71,333	2×10^{-4}	20	44,600
Box Packing	185	47,859	2×10^{-4}	25	37,400
Cup Insertion	170	61,814	2×10^{-4}	20	38,640

Deployment. At inference, observations are processed using the same image pipeline as during training. The instruction embedding is pre-computed and cached, so no text encoder is required on the robot. Actions are generated with 10 Euler steps in action-only mode; future-image prediction is disabled at inference. Predicted chunks are de-normalized using the corresponding training-set statistics and executed in a receding-horizon manner at 30 Hz.

D.2 FANUC CRX-10IA

Tasks and data. We evaluate two tabletop manipulation tasks: *Mug Hanging*, where the robot places a mug on a mug tree, and *Multi-stage Cooking*, which requires opening a pot, placing corn into the pot, and transferring garlic onto a plate. Demonstrations are collected by human teleoperation using a 6-DoF SpaceMouse at 10 Hz and converted to LeRobot v2.1. A separate model is trained for each task.

Observations and actions. A fixed third-person RealSense camera and a wrist-mounted RealSense camera provide two 240×240 RGB views. Each image is resized to 224×224 and the pair is concatenated into a 224×448 input (392 V-JEPA tokens). The action space is the 8-dimensional SpaceMouse command

$$[x, y, z, \mathrm{pitch}, \mathrm{roll}, \mathrm{yaw}, \mathrm{btn}_1, \mathrm{btn}_2],$$

while proprioception consists of the commanded Cartesian pose and gripper angle. Actions and states are min–max normalized. Because the action representation is incremental, padded action steps are set to zero. Each prediction contains 16 actions, corresponding to 1.6,s at 10,Hz.

Task	Episodes	Frames	LR	Epochs	Steps
Mug Hanging	136	25,124	1×10^{-4}	35	27,510
Cooking Sequence	81	19,936	1×10^{-4}	50	31,150

Deployment. The policy operates at 10,Hz using 10 Euler denoising steps in action-only mode. At each planning step, the first 8 actions of the 16-step prediction are executed before re-observation and replanning. Text embeddings and normalization statistics are loaded from the training checkpoint. Episodes are limited to 1000 control steps, and checkpoints are validated by open-loop replay on held-out demonstrations before real-world deployment.