LA-JEPA: COMPACT ROBOT POLICIES WITH LATENT WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

World-action models commonly adapt pretrained video generators, and vision-language-action models build on pretrained vision-language backbones; both tie policy architecture and inference cost to large transformers designed for generation rather than control. While the knowledge accumulated through large-scale pretraining is certainly useful, it remains unclear how much generation-oriented architectural choices affect control performance, and whether the trade-off required to access this pretraining is worth it. In this work, we introduce **Latent Action JEPA (La-JEPA)**, a compact world-action model whose policy transformer is trained from scratch in the feature space of a pretrained V-JEPA 2.1 encoder, so that its architecture and capacity can be chosen for control. Following Fast-WAM, La-JEPA learns to predict future features jointly with actions during training and generates only actions at inference. **Among policies without embodied pretraining, La-JEPA achieves state-of-the-art clean and randomized success on RoboTwin 2.0 and is equivalent to the state of the art on LIBERO-Plus, while using far fewer parameters than the strongest competing policies.** It runs substantially faster than Fast-WAM at comparable performance and outperforms the closest JEPA-based methods, LeapBot-WA and JEPA-WAM, on LIBERO-Plus and RoboTwin 2.0 without their pretrained policy backbones. On four real-robot tasks, including contact-heavy pushing and long-horizon multi-step manipulation, its average success is comparable to $\pi_{0.5}$, outperforming non-embodied-pretrained baselines. Through a design study of the choices behind La-JEPA, we show that JEPA features outperform video VAE latents, most clearly out of distribution, and that training-time future prediction choices significantly impact the policy even when future prediction is skipped at inference. These results establish a compact, efficient foundation for JEPA-based world-action modeling: without a generative pretrained policy backbone, La-JEPA is comparable to or better than far larger policies built on video and language generation models, suggesting that robot foundation models may benefit from architectures designed for control rather than inherited from generation.

1 INTRODUCTION

World-action models increasingly adapt pretrained video-generation transformers to robot control, inheriting useful visual and temporal priors along with the representation space, tokenization, and architecture of a model built for video synthesis (Yuan et al., 2026b; Ma et al., 2026; Kim et al., 2026). Vision-language-action policies take a similar route using language-generation transformers aligned with visual inputs (Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). The success of these two types of architectures raises two practical questions: must robot policies necessarily inherit the architecture of a generative backbone to be competitive, and can a compact world-action transformer trained from scratch actually reach the performance of these policies?

Most world-action models (WAMs) are VAE-based because they build on pretrained video generators, yet better-suited latent spaces are available. Video VAEs retain accurate information but spend much of their capacity on pixel-level fidelity that control does not require. JEPA representations, in contrast, are developed and evaluated for downstream tasks such as video understanding, depth estimation, and segmentation, which are closer to the needs of embodied agents (Assran et al., 2025; Mur-Labadia et al., 2026). Previous work aligns VLMs or video-generation models to the JEPA

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

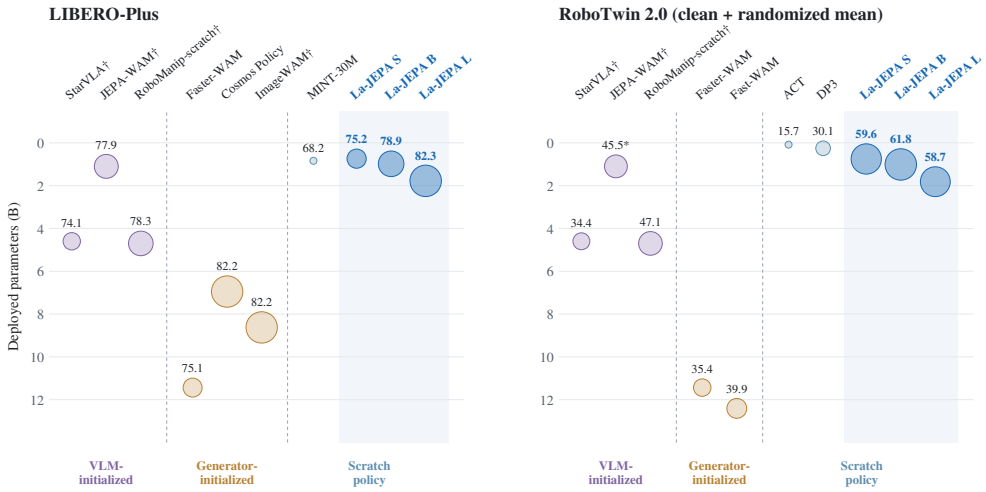


Figure 1: Success of policies without embodied pretraining on LIBERO-Plus and RoboTwin 2.0; the RoboTwin panel reports the equally weighted mean of clean and randomized success for clean-trained policies. Higher positions on the linear axis indicate fewer deployed parameters, including frozen encoders; diameters and labels give success in %. Groups identify the policy transformer's initialization: VLM and video/image groups inherit generatively pretrained transformers, whereas scratch policies do not. †: estimated parameter count; $*$: success estimated from JEPA-WAM's 20-task results (Appendix B). Fast-WAM (Table 1) is omitted from the LIBERO-Plus panel.

space (Lin et al., 2026; Liu et al., 2026) or learns a JEPA-style latent space jointly with the policy (Vujinovic & Kovacevic, 2025); to our knowledge, no prior work trains a compact world-action policy from scratch in a pretrained JEPA feature space.

We introduce **Latent Action JEPA (La-JEPA)**, a world-action model that learns future-feature prediction and action generation directly in the dense spatial and temporal feature space of a pretrained V-JEPA 2.1 video encoder (Mur-Labadia et al., 2026). Its World DiT and Action DiT are initialized from scratch, allowing their architecture and capacity to be chosen independently of the visual encoder. Following Fast-WAM and Faster-WAM (Yuan et al., 2026b; Ma et al., 2026), actions attend to observed context but not to future tokens, so predictive supervision can shape the policy during training without the cost of generating the future at inference time.

Among policies without embodied pretraining, La-JEPA achieves the strongest clean and randomized RoboTwin 2.0 results and is equivalent to the state of the art on LIBERO-Plus with four to five times fewer deployed parameters, while running up to 5$\times$ faster than Fast-WAM, whose design we follow (Figure 1, Table 1). It also outperforms the closest JEPA-based methods, LeapBot-WA and JEPA-WAM, on LIBERO-Plus and RoboTwin 2.0 (Table 2). On real-robot tasks ranging from cube placement to contact-rich pushing and long-horizon manipulation, La-JEPA trained from scratch on the target demonstrations achieves the highest average success among the compared policies, comparable to the embodied-pretrained $\pi_{0.5}$ (Table 3). Together, these results demonstrate leading benchmark performance, efficient inference, and competitive physical manipulation without requiring a generative backbone.

To understand which design choices enable this performance, we study the impact of the representation space, different targets for predictive supervision, and robustness mechanisms. We adopt a *compact first* approach, making the architecture viable even for a policy with 0.1B trainable parameters, before increasing policy capacity. Comparisons of JEPA features and VAE latents span both benchmarks and multiple model configurations, showing that JEPA features generally outperform VAE latents, with the clearest gains for out of distribution evaluations (Section 4.1, Appendix A.1). Action-only controls separately confirm the benefit of WAM-style future prediction, and studies of temporal encoding, dropout, augmentation, and encoder adaptation complete the recipe. These analyses support the central contribution: a compact, effective world-action policy, together with evidence to guide its design.

2 Related Work

Vision-language-action policies. Vision-language-action (VLA) policies extend pretrained vision-language models with action outputs, predicting discretized action tokens (Kim et al., 2024) or attaching a flow-matching action expert (Black et al., 2024), and broaden their training data to generalize further (Physical Intelligence et al., 2025; Yuan et al., 2026a). All inherit the size and inference cost of a multi-billion-parameter vision-language backbone.

World-action models. World-action models (WAMs) instead build on pretrained video or image generators whose representations drive action generation (Kim et al., 2026; Zhang et al., 2026), reaching the state of the art on LIBERO-Plus among policies without embodied pretraining. Because generating future frames slows inference, recent WAMs predict the future only during training (Yuan et al., 2026b) and reduce the action expert to a single layer that reads keys and values cached from all layers of the video backbone (Ma et al., 2026). All retain the generator backbone in the deployed policy.

Joint-embedding predictive architectures. Joint-embedding predictive architectures (JEPAs) learn representations by predicting the embeddings of masked inputs rather than reconstructing pixels (Assran et al., 2023). V-JEPA extends this objective to video (Bardes et al., 2024), V-JEPA 2 scales it (Assran et al., 2025), and V-JEPA 2.1 improves the dense features of V-JEPA 2 (Mur-Labadia et al., 2026), which makes it especially well suited to robotic policies. JEPA representations have since been used in robot policies: ACT-JEPA learns a JEPA-style latent space jointly with the policy (Vujinovic & Kovacevic, 2025), VLA-JEPA and JEPA-VLA bring V-JEPA targets or features into vision-language-action models (Sun et al., 2026; Miao et al., 2026), and JEPA-WAM and LeapBot-WA build world-action models on V-JEPA features (Lin et al., 2026; Liu et al., 2026). These two world-action models are the closest to La-JEPA, but both initialize their predictors from pretrained vision-language or video generation models.

Compact policies. Policies that are competitive without inheriting either a VLM or video backbone are scarce. While Diffusion Policy (Chi et al., 2023), DP3 (Ze et al., 2024) and ACT (Zhao et al., 2023) exist, their performance is rarely close to VLAs and World Models. MINT predicts coarse-to-fine action tokens with a separately trained action tokenizer (Huang et al., 2026) and reaches much better success rates, but remains below the strongest WAMs and VLAs. La-JEPA fits in this category as it trains its policy DiTs from scratch while adding WAM-style future prediction in a pretrained JEPA feature space.

3 La-JEPA

La-JEPA learns action generation jointly with future-feature prediction in a pretrained V-JEPA 2.1 ViT-L representation space, following the separation of training-time world modeling from inference-time action generation in Fast-WAM (Yuan et al., 2026b) and Faster-WAM (Ma et al., 2026). At decision time t, the policy maps multi-camera observations o_t, proprioception s_t, and a language instruction ℓ to an action chunk $A = (a_t, \dots, a_{t+H-1})$. This section describes the fixed architecture and training procedure (Figure 2); Section 4 examines the choices within it.

3.1 Architecture

Visual representations. The pretrained V-JEPA 2.1 ViT-L encoder (Mur-Labadia et al., 2026) processes each camera independently, producing observation features z_{ctx} from the most recent frames, and future targets z_{fut} representing the 32 frames ahead. We normalize and spatially reduce these feature grids before projecting them into the World DiT's embedding space. They are the model's only visual input and prediction target, so the model gets no separate RGB pathway. Future predictions are matched to targets directly in V-JEPA space and require no pixel reconstruction.

Language and proprioception. A frozen FLAN-T5-large encoder (Chung et al., 2022) embeds the instruction ℓ, and learned projections map proprioception s_t to additional tokens; together they form the conditioning sequence z_{cond}, which both DiTs read through cross-attention.

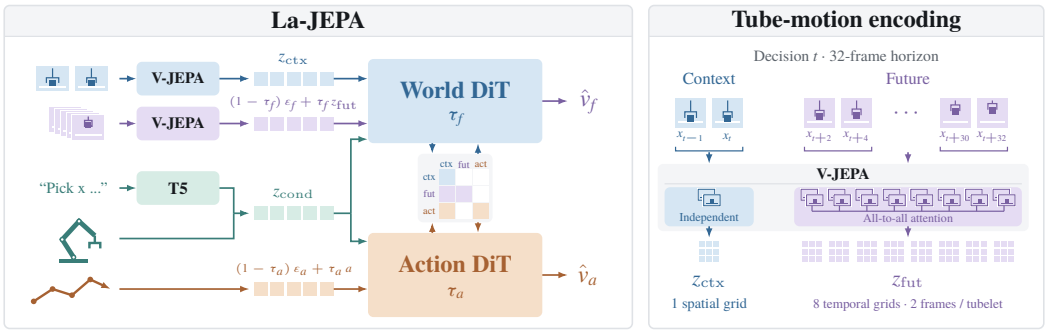


Figure 2: La-JEPA. Left: general architecture. Right: context and future encoding with V-JEPA 2.1.

World–action architecture. The model is made of two diffusion transformers (DiTs) (Peebles & Xie, 2022): a World DiT over observation and future-feature tokens and an Action DiT over the action chunk. Both are initialized from scratch with separate attention projections and feed-forward layers, so pretrained knowledge enters only through the visual and language encoders. Rotary visual positional encodings (Su et al., 2021) preserve spatial and temporal information, supplemented by camera-pose conditioning (Yuan et al., 2026a). The two DiTs interact at every layer, where action queries read the World DiT's observation keys and values alongside its own. The attention mask keeps future information out of the action pathway: observation tokens attend only to observations, future tokens to observations and futures, and action tokens to observations and actions. Actions therefore never access future features directly, yet predictive supervision still shapes the observation representations they read through the shared World DiT parameters during training.

3.2 Training and Inference

Joint flow matching. We train both DiTs with future and action-space flow matching (Lipman et al., 2022). For each training example, we independently sample timesteps $\tau_f, \tau_a \sim \mathcal{U}(0,1)$ and Gaussian noise ϵ_f, ϵ_a, forming the noisy inputs

$$z_{\text{fut}}^{\tau_f} = (1-\tau_f)\epsilon_f + \tau_f z_{\text{fut}}, \qquad A^{\tau_a} = (1-\tau_a)\epsilon_a + \tau_a A. \tag{1}$$

The model receives $(z_{\text{ctx}}, z_{\text{fut}}^{\tau_f}, A^{\tau_a}, z_{\text{cond}})$ and predicts velocities $\hat{v}_f$ and $\hat{v}_a$ under the attention mask as described above. The World DiT is conditioned on τ_f and the Action DiT on τ_a. The training objective is

$$\mathcal{L} = \mathbb{E}\left[\text{MSE}(\hat{v}_f, z_{\text{fut}} - \epsilon_f) + \text{MSE}(\hat{v}_a, A - \epsilon_a)\right], \tag{2}$$

where each mean squared error averages over its supervised elements.

Action inference. At inference, we drop the future-feature tokens, initialize the action chunk from Gaussian noise, and integrate the action velocity field from $\tau_a = 0$ to $\tau_a = 1$ with Euler steps, holding the World DiT's timestep conditioning at a fixed value $\tau_{\text{ctx}} = 0$ (since we do not predict the futures anyway). Since observation tokens attend only to one another under fixed conditioning, their per-layer keys and values are computed once and cached, and only the Action DiT runs at each denoising step. Like most recent policies, we convert the predicted action tokens to the robot's action space, execute the first k actions of the chunk ($k = 10$ on LIBERO and 24 on RoboTwin, as in Fast-WAM), and then predict a new chunk from the latest observations.

4 Design Analysis

Experimental setup. We test the design choices behind Section 5 with the small (S) policy unless stated otherwise. We ask, in turn, whether to predict future features, in which feature space, how to encode motion (Section 4.1), and then how to make the policy robust to observation changes (Section 4.2). The studies use LIBERO, LIBERO-Plus, and clean-trained RoboTwin 2.0 (Liu et al., 2023; Fei et al., 2025; Chen et al., 2025); in-distribution (IID) evaluation refers to LIBERO or clean RoboTwin scenes, and out-of-distribution (OOD) evaluation to LIBERO-Plus or randomized scenes.

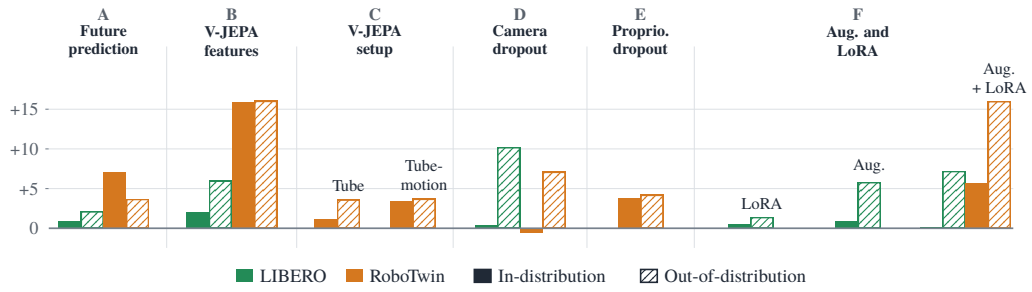


Figure 3: Gain in success (percentage points) from each design choice over its baseline: A, action-only training; B, Wan2.2 VAE latents; C, static encoding; D, no camera dropout (LIBERO: wrist-only dropout); E, no proprioception dropout; F, a plain policy without augmentation or encoder LoRA. Green: LIBERO; orange: RoboTwin 2.0; solid: in-distribution; hatched: out-of-distribution. Block A averages LIBERO-Plus gains over three training seeds. Values and evaluation budgets in Table 4.

Figure 3 reports each choice's gain in success in percentage points (hereafter, points), grouped into blocks A to F. Each block has its own reference recipe, so comparisons hold within a block; Table 4 lists baselines and evaluation budgets, and Appendix A documents the protocols.

4.1 Prediction Target and Feature Space

Predictive supervision. We first test whether predicting future features benefits the policy, comparing joint action and future-feature prediction with action-only training (Figure 3A). Future prediction raises RoboTwin success by 7.0 points in clean and 3.7 in randomized scenes. On LIBERO-Plus, it improves success for each of three training seeds, by 2.1 points on average, with every paired comparison significant ($p < 0.01$, McNemar; Appendix A.1).

Feature space. We next ask in which space prediction should be learned, comparing frozen V-JEPA 2.1 features with latents of the VAE of Wan2.2 (Team Wan et al., 2025), the video model underlying Fast-WAM, Faster-WAM, and LeapBot-WA (Figure 3B). Each policy learns action and future-feature prediction with the normalization and projections suited to its representation, so the comparison evaluates each representation together with its interface to the policy. V-JEPA improves success in all four evaluations of this pair: by 15.9 points in clean and 16.0 in randomized RoboTwin scenes, and by 6.0 on LIBERO-Plus and 2.0 on LIBERO. In randomized scenes, the Wan2.2 policy collapses and succeeds in only 1.0% of episodes, close to the video-generator WAMs in Table 1. In two additional RoboTwin comparisons (Appendix A.1), clean success is comparable in one and favors Wan2.2 in the other, whereas randomized success, reported for the second, again strongly favors V-JEPA. These results echo those of LDA-1B, which similarly reports a large gain from replacing VAE latents with DINO features (Lyu et al., 2026).

Temporal encoding. Unlike LDA-1B's DINO, which only encodes images, V-JEPA 2.1 can process videos, which raises the question of how the observations and future targets should be encoded. We tested three configurations. *Static* encodings treat every frame as an image: the observation is the current frame, and eight future targets, spaced four frames apart up to 32 frames ahead, are encoded separately. *Tube* encoding instead processes two-frame clips aligned with the encoder's tubelets, ending at the current frame for the observation and at each target frame for the futures; the observation clip thus gives the policy a motion cue without access to future observations. *Tube-motion* keeps the tube observation but jointly encodes the 16 future frames, spaced two frames apart, so each of the eight future feature grids attends over the whole predicted horizon (Figure 2, right). Relative to static encoding on RoboTwin (Figure 3C), tube encoding raises clean success by 1.1 points and randomized success by 3.6, and tube-motion preserves the randomized gain (+3.7) while raising the clean gain to 3.4; the final recipe therefore uses tube-motion.

Token reduction. Before entering the World DiT, the encoded feature grids are reduced spatially, either by averaging neighboring features (mean pooling) or by concatenating them into one token

Table 1: Policies without embodied pretraining (all use pretrained perception). Parameters in billions; La-JEPA sizes for LIBERO. Plus: episode-weighted LIBERO-Plus; C2C/C2R: clean-trained RoboTwin in clean/randomized scenes. Success in %; best in bold.

		Params (B)		LIBERO		RoboTwin 2.0		
Method	Transformer init.	Train.	Depl.	IID	Plus	C2C	C2R	Mean
Qwen-RobotManip-scratch	VLM (supervised)	4.7	4.7	98.2	78.3	71.6	22.6	47.1
StarVLA	VLM (supervised)	4.6	4.6	98.0	74.1	58.1	10.6	34.4
Fast-WAM	video generator	6.0	12.4	97.6	49.7	77.8	1.9	39.9
Faster-WAM	video generator	5.1	11.4	**98.5**	75.1	68.6	2.2	35.4
Cosmos Policy	video generator	2.0	7.0	**98.5**	82.2	—	—	—
ImageWAM	image generator	4.5	8.6	98.4	82.2	—	—	—
JEPA-WAM	VLM (supervised)	0.3	1.1	96.7	77.9	—	—	—
La-JEPA S (ours)	scratch	0.10	0.74	93.4	75.2	**80.2**	39.0	59.6
La-JEPA B (ours)	scratch	0.34	0.98	96.4	78.9	78.8	**44.7**	**61.8**
La-JEPA L (ours)	scratch	1.14	1.78	97.0	**82.3**	78.4	39.1	58.7

(patch grouping). We initially used mean pooling for its simplicity, since it can be applied once to the precomputed features. On RoboTwin, patch grouping improved randomized success, so the final RoboTwin recipe uses it; LIBERO did not benefit from it (89.35% against 94.05% with mean pooling), so we kept mean pooling there (Appendix A.1).

4.2 Robustness and Encoder Adaptation

Camera dropout. To reduce reliance on any single view, we drop eligible cameras during training, always retaining at least one view and removing the dropped cameras' future targets from the loss (Figure 3D). This is the condition dropout used to train classifier-free guidance (Ho & Salimans, 2022), although we sample without guidance. Dropping only the wrist cameras raises randomized RoboTwin success by 7.1 points, while clean success remains essentially unchanged. LIBERO benefits from this too: by allowing either camera to be dropped, we raise LIBERO-Plus success by 10.2 points, with large gains under camera-viewpoint and sensor-noise perturbations (Table 6), while in-distribution success slightly improves.

Proprioception dropout. Dropping the proprioceptive conditioning with probability 0.1 during training, with the configuration otherwise unchanged, raises RoboTwin success by 3.8 points in clean and 4.2 in randomized scenes (Figure 3E). The final recipe therefore uses both camera and proprioception dropout.

Data augmentation and encoder adaptation. Finally, we combine image augmentation of the observations with LoRA (Hu et al., 2021) adaptation of the observation encoder; future targets are still computed from unmodified frames by the original encoder for training efficiency purposes (Appendix B). Relative to a plain LIBERO policy (Figure 3F), LoRA alone adds 1.4 points on LIBERO-Plus, augmentation alone 5.8, and both together 7.2, while in-distribution success remains unchanged. The combined recipe also improves RoboTwin, raising clean success by 5.7 points and randomized success by 16.0 (Appendix A.2).

4.3 Final Recipe

These studies define the final recipe: joint action and future prediction in V-JEPA 2.1 tube-motion features, camera and proprioception dropout, and image augmentation with encoder LoRA. We train it at three sizes, S, B, and L (Table 7), which correspond to 0.10, 0.34, and 1.14 B trainable parameters respectively. Appendix B documents the remaining benchmark-specific settings.

Table 2: Latent world-action models. Embodied PT: pretraining on robot data beyond the target demonstrations. ‡Reported by Yan et al. (2026).

			Params (B)		LIBERO		RoboTwin 2.0 (20 tasks)	
Method	Transformer init.	Embodied PT	Train.	Depl.	IID	Plus	C2C	C2R
LDA-1B‡	—	yes	—	—	90.6	45.5	—	—
LeapBot-WA	video generator	yes	—	2.9	97.3	71.1	—	—
JEPA-WAM	VLM (supervised)	no	0.3	1.1	96.7	77.9	79.9	36.9
La-JEPA S (ours)	scratch	no	0.10	0.74	93.4	75.2	92.5	50.5
La-JEPA B (ours)	scratch	no	0.34	0.98	96.4	78.9	93.0	59.5
La-JEPA L (ours)	scratch	no	1.14	1.78	97.0	82.3	92.0	54.2

5 Experimental results

Setup. We evaluate the final recipe against WAM and VLA policies that, like La-JEPA, use no embodied pretraining on robot data beyond the target demonstrations (Figure 1, Table 1), and against the closest latent world-action models (Table 2); we then compare computational cost and, on four physical tasks, real-robot success (Table 3). Model sizes distinguish the parameters updated on target demonstrations (*trainable*) from the full inventory including frozen encoders (*deployed*; Appendix C.2).

5.1 Benchmark Performance and Generalization

Both benchmarks test generalization beyond the training distribution. On RoboTwin, policies are trained on clean demonstrations only and evaluated in both clean scenes and randomized scenes, which add clutter, distractor objects, and new textures. On LIBERO, policies are trained on the four standard suites and evaluated both on them and on LIBERO-Plus, which perturbs them along seven axes such as camera viewpoint, lighting, and sensor noise. On RoboTwin, all three La-JEPA sizes outperform every baseline in both regimes, with the largest margin in randomized scenes (Table 1). On LIBERO-Plus, success increases with size, and L is equivalent to the best baselines, Cosmos Policy and ImageWAM, which deploy four to five times more parameters. On LIBERO, where most methods are close to saturation, L comes within 1.5 points of the best baseline.

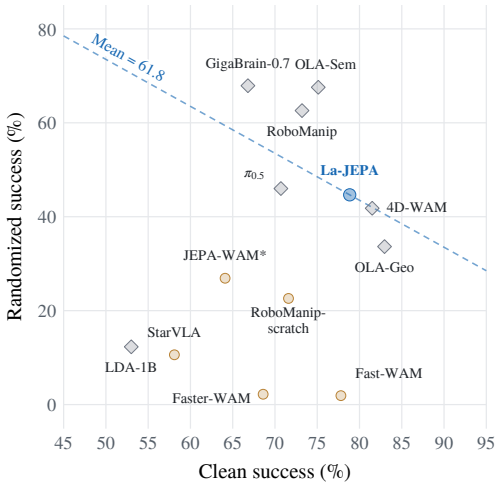


Figure 4: RoboTwin clean versus randomized success of clean-trained policies. Blue: La-JEPA B; brown: no embodied pretraining; gray: embodied pretraining; RoboManip: Qwen-RobotManip. Points above the dashed line through La-JEPA have a higher mean of the two regimes. ∗: JEPA-WAM estimate (Appendix B); LDA-1B as reported by Yan et al. (2026).

RoboTwin success, in contrast, does not monotonically increase with size. We did not vary the relative size of the Action DiT, which matches the World DiT at every size, and we hypothesize that a larger action expert more easily overfits the action distribution, which is far simpler than the distribution of future features, even when regularized by the world loss. Faster-WAM, whose single-layer action expert generalizes well on LIBERO-Plus (Ma et al., 2026), is consistent with this hypothesis.

Figure 4 adds embodied-pretrained policies for reference. La-JEPA B's mean over the two regimes exceeds that of all but three of them, OLA-Sem, GigaBrain-0.7 (GigaBrain Team et al., 2026), and Qwen-RobotManip, which all reach higher randomized but lower clean success, as expected when robot pretraining data already covers part of the randomized variation.

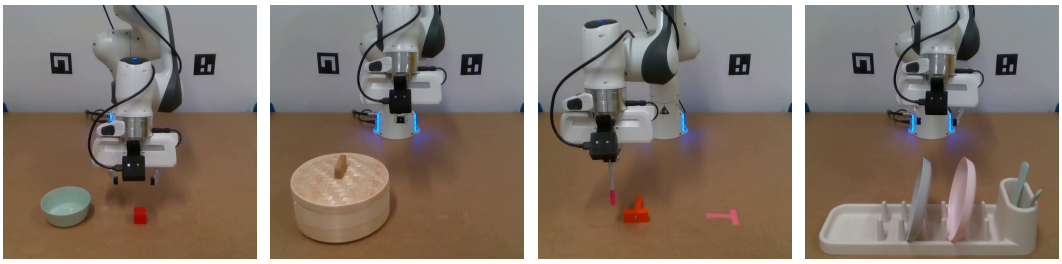


Figure 5: Real-robot tasks: cube placement, fruit picking, Push T, and dish manipulation.

5.2 Comparison with Latent World-Action Models

La-JEPA also outperforms the two closest JEPA-based world-action models on LIBERO-Plus and RoboTwin 2.0, although both retain pretrained predictors (Table 2). LeapBot-WA, which also pretrains on robot data, falls below all three La-JEPA sizes on LIBERO-Plus. JEPA-WAM, built on a pretrained Qwen2.5 model (Qwen et al., 2024), is exceeded on LIBERO-Plus by our B and L models, the former deploying fewer parameters. On RoboTwin, JEPA-WAM trains and evaluates on 20 of the 50 tasks, and on these tasks every La-JEPA size, trained on all 50, leads it by more than ten points in both regimes. This subset is easier than the full benchmark, since per-task DP3 and ACT policies score about 20 points higher on it in clean scenes (Appendix B); because it was chosen by JEPA-WAM and applies equally to both methods, we do not consider the comparison biased in our favor. Even when JEPA-WAM's scores are extrapolated to all 50 tasks using this difficulty gap, every La-JEPA size remains ahead in both regimes. LDA-1B, pretrained on over 30k hours of human and robot data with a 4B vision-language model (Lyu et al., 2026), remains far below on both benchmarks (Yan et al., 2026).

5.3 Computational Cost

La-JEPA updates 0.10 to 1.14 B parameters during training, far fewer than the video-generator and vision-language-action baselines (2.0 to 6.0 B; Table 1) while showing lower replan latency at matched or better performance. We time complete policy calls, with ten denoising steps on one consumer GPU (Table 9, Appendix C). With optimized (compiled) builds, S and B run two to five times faster than Fast-WAM and Faster-WAM, and S remains 2.6× faster than optimized Fast-WAM with original-kernel CUDA graphs, which reproduce uncompiled actions bit-for-bit. Compiled builds do not significantly change closed-loop LIBERO success (Appendix C), and the physical-robot policy runs compiled. Reducing inference to three denoising steps lowers latency further while keeping success within about one point (Appendix E.2).

Table 3: Real-robot successes out of 20 and benchmark policy-call latency in ms, not control-loop timing (Appendix C); LingBot-VA's is rounded to 10 ms. Position: full dish success with correct utensil placement; Average: mean over the four tasks. LingBot-VA dish: 10 trials. Bold: best.

Policy	Latency (ms)	Fruit	Cube	Push T	Dish manipulation				Average
					Plates	Utensil 1	Utensil 2	Position	
With embodied pretraining									
$\pi_{0.5}$	71.89	**20**	**20**	11	**20**	**19**	**19**	**18**	17.25
LingBot-VA	≈ 4760	18	**20**	2	6/10	3/10	2/10	1/10	—
Without embodied pretraining									
Fast-WAM	88.09	19	19	14	**20**	**19**	15	12	16.00
Faster-WAM	50.96	18	**20**	13	**20**	18	16	14	16.25
La-JEPA	25.16	19	**20**	**15**	**20**	18	17	16	**17.50**

5.4 Real-Robot Experiments

We train a La-JEPA B policy from scratch, with encoder LoRA but without image augmentation, on demonstrations of four Franka tabletop tasks (Figure 5; Appendix D) and compare it with Fast-WAM, Faster-WAM, $\pi_{0.5}$, and LingBot-VA (Li et al., 2026) in Table 3. We start with a simple cube placement as control, then test on fruit picking which requires lifting a lid, setting it down, and choosing the correct fruit; Push T tests contact-heavy interactions; and our dish task requires placing two plates from a rack onto the table, then a spoon on the gray plate and a fork on the pink plate in one uninterrupted rollout, with interchangeable utensil positions that require distinguishing the utensils.

La-JEPA achieves the highest average success among policies without embodied pretraining and is comparable to the embodied-pretrained $\pi_{0.5}$, although its policy transformer is trained only on the target demonstrations. It is the most successful policy on Push T, whereas $\pi_{0.5}$ leads on the long-horizon dish task. On successful trials, La-JEPA also completes every task faster than the other policies without embodied pretraining (Appendix D), and faster than $\pi_{0.5}$ on fruit picking and Push T, though slower on the dish task. LingBot-VA rarely succeeds on Push T and dish manipulation and is several times slower than every other policy. LingBot-VA's low success may reflect a need for more real-robot finetuning than the shared budget allowed (Appendix D).

6 Limitations and Future Work

Scope of recipe optimization. We aim to show the potential of compact world-action policies rather than to optimize them exhaustively: beyond the targeted design studies, we did not systematically search learning rates, weight decay, dropout rates, LoRA configurations, or the relative capacity of the World and Action DiTs. This lack of scale-specific tuning, together with limited target data, may contribute to the non-monotonic RoboTwin scaling; our experiments do not isolate these effects. We also recognize that many high-performing baselines have external supervision objectives and robot-specific architectures that we do not: while this work does outline the performance of a policy with a simple setup, we cannot be certain that our improvements would stack with other work's policy upgrades.

Large-scale pretraining. Our policies learn from target demonstrations alone, so how La-JEPA scales with embodied pretraining remains open. Adapting the policy and its visual representation across embodiments and environments would also test whether data, rather than capacity, limits the larger models. We consider this the principal future direction of research that should come out of this paper. La-JEPA remains below the strongest embodied-pretrained methods on randomized RoboTwin scenes, and its real-robot average, although comparable to that of $\pi_{0.5}$, comes from four tasks on one robot setup and does not establish parity across broader manipulation demands. The richer supervision, synthetic data, and additional objectives used by some stronger baselines are natural extensions of La-JEPA that we have not tested.

7 Conclusion

We asked whether world-action policies must inherit the architecture of a generative backbone, and whether a compact transformer trained from scratch can compete with those that do. Trained in a pretrained JEPA feature space, La-JEPA reaches the strongest RoboTwin 2.0 clean and randomized results among policies without embodied pretraining, and LIBERO-Plus performance equivalent to the best reported among comparable baselines, while running up to 5$\times$ faster than Fast-WAM. On four physical tasks, La-JEPA's average success is comparable to that of the embodied-pretrained $\pi_{0.5}$. Through our study, we empirically validate the benefits of the V-JEPA latent space for control tasks, provide some guidance on the benefits of different robustness design improvements and find that heavy generative models are not a requirement for high-performing learned policies.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, including the optimization of the training and dataloading pipeline which was particularly heavy, evaluation, and data-preprocessing code; to reformat datasets into precomputed feature stores (always through verifiable scripts); and to help design the infrastructure that enabled the interpretation of our large amount of results. We did not use generative AI tools to generate synthetic data, develop theoretical models, formulate or prove mathematical claims, or translate text; these uses are not applicable to this work. Additionally, we used generative AI tools to edit parts of the paper, to write the code that generates figures and tables from recorded results, to search for and summarize related literature, and to format references. We have reviewed all AI-assisted work: code was tested and validated against recorded outputs, reported numbers were traced to their evaluation records, figures and tables are generated by code from the underlying data directly, never by the AI tool itself, and references were checked against their arXiv entries. 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 uses public simulation benchmarks and robot demonstrations collected by the authors through teleoperation; it involves no human-subject studies and releases no new dataset. We do not foresee harms specific to this work beyond those of robot learning in general.

REPRODUCIBILITY STATEMENT

Anonymized code for feature precomputation, training, and LIBERO, LIBERO-Plus, and RoboTwin 2.0 evaluation, with the configurations of the final S, B, and L policies, is provided in the supplementary material. Section 3 specifies the architecture and training objective. Appendix A details the design studies, Appendix B the training configurations, evaluation protocols, and baseline sources, Appendix C the latency and parameter accounting, Appendix D the real-robot setup, and Appendix E supplementary results, including per-task evaluations.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture, 2023. arXiv:2301.08243.

Mido 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, 2025. arXiv:2506.09985.

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, 2024. arXiv:2404.08471.

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, 2024. arXiv:2410.24164.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan

Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation, 2025. arXiv:2506.18088.

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, 2023. arXiv:2303.04137.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling Instruction-Finetuned Language Models, 2022. arXiv:2210.11416.

Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. UniMax: Fairer and more Effective Language Sampling for Large-Scale Multilingual Pretraining, 2023. arXiv:2304.09151.

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, 2025. arXiv:2510.13626.

Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. RankMe: Assessing the downstream performance of pretrained self-supervised representations by their rank, 2022. arXiv:2210.02885.

GigaBrain Team, Angen Ye, Axiang Sun, Can Jin, Chenxi Cheng, Chong Shi, Dengke Shang, Dingqian Zhang, Guan Huang, Guangqiang Wang, Guangqing Ding, Guo Li, Hangcong Li, Hengyu Zhong, Hongtao Lu, Jianbo Qin, Jiming Mao, Jing Zhu, Jindi Lv, Jingzhi Cui, Junjie Xie, Junyi Bao, Kai Liu, Lei Yuan, Limin Long, Lv Feng, Mingming Yu, Peng Li, Pengfei Yi, Qi Li, Qianli Zhang, Qingfang Li, Qitang Hu, Rui Zhang, Shaoyan Sun, Shibo Sun, Shiying Duan, Tenghui Chen, Tianze Liu, Weijie Ke, Wenyao Xue, Xiaofeng Wang, Xiaoyu Tian, Xinyu Liu, Xinze Chen, Yang Wang, Yankai Wang, Yejun Zeng, Yifan Li, Yifei Nie, Yilong Li, Yilong Liu, Yongchao Feng, Yumeng Wang, Yun Ye, Zhichao Liu, Ziheng He, Zonghai Yang, and Zheng Zhu. GigaBrain-0.7: Scaling Embodied Foundation Models to Emergent Capabilities with a Three-System Architecture, 2026. arXiv:2608.15875.

Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance, 2022. arXiv:2207.12598.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021. arXiv:2106.09685.

Renming Huang, Chendong Zeng, Wenjing Tang, Jintian Cai, Cewu Lu, and Panpan Cai. Mimic Intent, Not Just Trajectories, 2026. arXiv:2602.08602.

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, 2024. arXiv:2406.09246.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning, 2026. arXiv:2601.16163.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control, 2026. arXiv:2601.21998.

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, 2026. arXiv:2608.09381.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling, 2022. arXiv:2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning, 2023. arXiv:2306.03310.

Pei Liu, Nan Zheng, Lang Zhang, Daojie Peng, Yanan Zhang, Feilong Kong, Mingyue Feng, Jiachao Liu, Yaonong Wang, Qifeng Chen, and Jun Ma. LeapBot-WA: World-Anchor Action Models via Predictive Latent Alignments, 2026. arXiv:2607.23969.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a Diffusion Foundation Model for Bimanual Manipulation, 2024. arXiv:2410.07864.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, Wenbo Cui, Senmao Qi, Shuo Wang, Yixin Zheng, Mi Yan, Xuesong Shi, Haoran Li, Dongbin Zhao, Ming-Yu Liu, Zhizheng Zhang, Li Yi, Yizhou Wang, and He Wang. LDA-1B: Scaling Latent Dynamics Action Model via Universal Embodied Data Ingestion, 2026. arXiv:2602.12215.

Liheng Ma, Rui Heng Yang, Zhanguang Zhang, Mateo Clemente, Ziwen Hu, Tongtong Cao, and Yingxue Zhang. Faster-WAM: Do World Action Models Need Deep Action Modules?, 2026. arXiv:2608.02365.

Shangchen Miao, Ningya Feng, Jialong Wu, Ye Lin, Xu He, Dong Li, and Mingsheng Long. JEPA-VLA: Video Predictive Embedding is Needed for VLA Models, 2026. arXiv:2602.11832.

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, 2026. arXiv:2603.14482.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, 2022. arXiv:2212.09748.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization, 2025. arXiv:2504.16054.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report, 2024. arXiv:2412.15115.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics, 2025. arXiv:2506.01844.

StarVLA Community. StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing, 2026. arXiv:2604.05014.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding, 2021. arXiv:2104.09864.

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, 2026. arXiv:2602.10098.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and Advanced Large-Scale Video Generative Models, 2025. arXiv:2503.20314.

Aleksandar Vujinovic and Aleksandar Kovacevic. ACT-JEPA: Novel Joint-Embedding Predictive Architecture for Efficient Policy Representation Learning, 2025. arXiv:2501.14622.

Haodong Yan, Junfeng Li, Junjie He, Zhide Zhong, MingMing Yu, Wenxuan Song, Jiaguan Zhu, Yangyang Zheng, Yuqiao Du, Jiadi You, Yingjie Cai, Xu Yan, Guanyi Zhao, Bingbing Liu, and Haoang Li. Robust-WAM: Bridging Generative Pretraining and Semantic Foresight in World-Action Models, 2026. arXiv:2608.05903.

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, 2026. arXiv:2608.08023.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, Jie Zhang, Jingyang Fan, Gengze Zhou, Qihang Peng, Chenxu Lv, Xiaoyue Chen, An Yang, Fei Huang, Junyang Lin, Dayiheng Liu, Jingren Zhou, Chenfei Wu, and Xiong-Hui Chen. Qwen-RobotManip Technical Report: Alignment Unlocks Scale for Robotic Manipulation Foundation Models, 2026a. arXiv:2606.17846.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination?, 2026b. arXiv:2603.16666.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations, 2024. arXiv:2403.03954.

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?, 2026. arXiv:2606.19531.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, 2023. arXiv:2304.13705.

Table 4: Gain in success (points) of each design choice over its baseline; comparisons hold within a block.

Choice	Baseline	Benchmark	IID	OOD	Evaluation
A Future prediction					
Action + future prediction	Action only	LIBERO	+0.9	+2.1	IID: seed 42; OOD: 3 seeds
Action + future prediction	Action only	RoboTwin	+7.0	+3.7	5,000/cond., 120–150k
B V-JEPA features					
V-JEPA 2.1 features	Wan2.2 VAE latents	LIBERO	+2.0	+6.0	2,000 / 10,030, 150k
V-JEPA 2.1 features	Wan2.2 VAE latents	RoboTwin	+15.9	+16.0	5,000/cond., 150k
C V-JEPA setup					
Tube	Static	RoboTwin	+1.1	+3.6	1,000/cond., 120–150k
Tube-motion	Static	RoboTwin	+3.4	+3.7	1,000/cond., 120–150k
D Camera dropout					
Camera dropout	Wrist-only camera dropout	LIBERO	+0.4	+10.2	n=2,000 / 10,030, 90k
Camera dropout	No camera dropout	RoboTwin	−0.6	+7.1	1,000/cond., 150k
E Proprioception dropout					
Proprioception dropout	No proprioception dropout	RoboTwin	+3.8	+4.2	1,000/cond., best step
F Augmentation and encoder LoRA					
Encoder LoRA	Plain	LIBERO	+0.5	+1.4	n=2,000 / 10,030
Image augmentation	Plain	LIBERO	+0.9	+5.8	n=2,000 / 10,030
Augmentation + LoRA	Plain	LIBERO	+0.1	+7.2	n=2,000 / 10,030
Augmentation + LoRA	Plain	RoboTwin	+5.7	+16.0	5,000/cond., 150k

A Design Studies

This appendix documents the design studies of Section 4. It first lists the values behind Figure 3 and the setup shared by all studies, then gives the protocol of each comparison in the order of the main text.

Summary of gains. Table 4 lists the gains shown in Figure 3, together with each block's baseline and evaluation budget. Values are percentage-point differences in success between La-JEPA's choice and its baseline; IID and OOD denote LIBERO and LIBERO-Plus, or clean and randomized RoboTwin scenes. Each block has its own reference recipe and checkpoints, so comparisons hold only within a block.

Shared setup. All design-study transformers are trained from scratch and evaluated with EMA weights, including the corresponding EMA encoder weights for policies with encoder LoRA. Studies use the small (S) policy unless stated otherwise, and each result comes from a single training run, except for the three-seed LIBERO-Plus comparison of predictive supervision. Several studies train on a subset of the training demonstrations, nominally 95%, which on LIBERO retains 1,663 of the 1,743 episodes; we call it the *95% split* and state the training data of each comparison below. LIBERO evaluations cover 2,000 episodes with a 700-step budget in every suite unless stated otherwise, and LIBERO-Plus evaluations all 10,030 episodes with the upstream suite-specific step limits; both redraw action noise at every replan. RoboTwin evaluation budgets are listed per block in Table 4; the 1,000-episode budgets are ablation evaluations, smaller than the 5,000-episode headline protocol (Appendix B).

A.1 Prediction Target and Feature Space

These studies determine whether the policy should predict future features, in which representation, and how motion and spatial resolution enter the tokens (Section 4.1).

Predictive supervision on LIBERO. Joint action and future-feature prediction improves LIBERO-Plus success over action-only training for each of three training seeds, raising mean success from 74.35% to 76.47%, a gain of 2.12 points (Table 5). Figure 3A shows this mean gain;

Table 5: Predictive supervision across training seeds on LIBERO-Plus (success in %; per-seed exact McNemar tests).

Seed	Selected step	Joint	Action only	Gain (pp)	Paired p
42	150k	76.13	74.47	+1.67	2.3×10^{-4}
1	90k	76.13	74.77	+1.37	0.002
2	120k	77.15	73.82	+3.33	2.2×10^{-12}
Mean	—	76.47	74.35	+2.12	—

its in-distribution bar uses the seed-42 pair at 150k updates, evaluated on 2,000 LIBERO episodes. Both objectives train on all 1,743 LIBERO episodes for 150,000 updates with image augmentation and encoder LoRA, whereas the headline S uses the 95% split and is selected at 120k (Appendix B), so these controls are not directly comparable to Table 1. The action-only control omits future-feature supervision and computation while preserving the action pathway. For each objective and training seed, we select the checkpoint with the highest LIBERO success from the 30k, 60k, 90k, 120k, and 150k grid and evaluate it on all 10,030 LIBERO-Plus episodes with EMA weights, action-noise scale 1 redrawn at every replan, ten denoising steps, and ten-action execution chunks. Selection picks the same step for both objectives within each seed. Paired tests use identical task and evaluation-seed episodes for the two objectives.

Predictive supervision on RoboTwin. On 5,000 episodes per regime, each policy on its own set of evaluation episodes, joint training achieves 80.16% clean and 39.02% randomized success, against 73.14% and 35.34% for action-only training. Both policies use the final augmentation recipe (Table 8), all 2,500 clean demonstrations, 150,000 training updates, and a global batch of 400, split into four data-parallel ranks of 100 for joint training and eight of 50 for action-only training; the action-only policy neither loads nor computes future targets. Selection on the 1,000-episode clean grid picks 150k for joint training and 120k for action-only training, with ties broken toward the later step.

Feature space. Block B compares frozen, static V-JEPA 2.1 features with Wan2.2 VAE latents at 150k updates, with action and future-feature prediction on the 95% split. Within each benchmark, feature dimensions, normalization, and input and output projections remain representation-specific, so the comparison concerns the two representation systems rather than their pretraining objectives in isolation. On RoboTwin, both policies use 384-dimensional, 12-layer transformers with patch grouping for observations and future targets and are evaluated on 5,000-episode clean and randomized manifests with 24-action execution chunks. On LIBERO, both use the 768-dimensional, 12-layer B transformer with $2\times$ spatial mean pooling and 10-action execution chunks; LIBERO evaluation instead uses a 600-step budget per suite and 30 settling steps, and LIBERO-Plus evaluation uses action-noise scale 1. On LIBERO-Plus, V-JEPA succeeds in 5,773 episodes and Wan2.2 in 5,173. All evaluations use EMA weights, a single observation context, and 10 denoising steps.

Additional feature-space comparisons. Two further RoboTwin comparisons pool the future targets, each evaluated on 5,000 episodes per regime (clean only for the first). With full-data, static-feature training and patch-grouped observations, V-JEPA and Wan2.2 achieve 61.50% and 61.44% clean success at 150k. In the second, with 768-dimensional transformers, Wan2.2 leads on clean success (63.90% versus 61.12%), whereas V-JEPA leads on randomized success (15.00% versus 1.28%). V-JEPA's advantage therefore holds in randomized scenes but not consistently in clean ones.

Temporal encoding. Block C compares three frozen-encoder policies with patch grouping, trained on the 95% split, that differ only in the encoding: static, tube, or tube-motion. Each regime uses 1,000 episodes on the same scenes, and randomized success is read at the checkpoint selected on clean episodes: 150k for static and tube-motion, and 120k for tube.

Token reduction. On RoboTwin, grouping neighboring features into one token by concatenation (patch grouping) instead of averaging them (mean pooling) changes clean success by $+0.3$ points

Table 6: LIBERO-Plus success under camera-viewpoint and sensor-noise perturbations (%).

Training variant	Camera viewpoints	Sensor noise
Frozen encoder, stored features		
Wrist-camera dropout only	11.4	42.0
Either-camera dropout	43.8	67.1
Online encoder, either-camera dropout		
Image augmentation only	74.6	81.4
Encoder LoRA only	43.9	73.3
Image augmentation + encoder LoRA	72.5	83.6

and raises randomized success by 4.1 points. Both policies apply the reduction to two-frame observations and to motion-based future targets and are compared at 150k updates; each clean and randomized evaluation uses 1,000 episodes, 32 samples per pixel, rendering depth 8, OIDN denoising, and action noise fixed within each episode. The pooling policy uses the full training set and the patch-grouping policy the 95% split, and their target normalization and projections also differ, so the comparison does not isolate the reduction operator. By contrast, on LIBERO, in a one-variable comparison between stored-feature S policies with tube-motion encoding at 150k updates, evaluated identically on 2,000 episodes, mean pooling reaches 94.05% against 89.35% for patch grouping. The final recipe uses patch grouping on RoboTwin, reducing the head camera $2\times$ and the wrist cameras $4\times$, and $2\times$ mean pooling on LIBERO (Appendix B). Appendix E.3 reports a related sweep that pools only the future targets.

A.2 Robustness and Encoder Adaptation

These studies target robustness to changes in the observations through camera and proprioception dropout, image augmentation, and encoder adaptation (Section 4.2). All LIBERO variants in blocks D and F train on the 95% split and are selected on LIBERO success: stored-feature policies with a frozen encoder at 90k updates, and online-encoder variants at 120k.

Camera dropout on LIBERO. The LIBERO row of block D compares either-camera with wrist-only dropout in stored-feature policies. Allowing either camera to be dropped changes both camera eligibility and the expected number of missing streams, from 0.10 to 0.19 per example; at least one stream is always retained. Table 6 reports the effect of this change, and of the adaptation variants below, on the camera-viewpoint and sensor-noise categories of LIBERO-Plus.

Camera and proprioception dropout on RoboTwin. The RoboTwin row of block D compares two frozen-encoder, static-feature policies at 150k updates that differ only in wrist-camera dropout with probability 0.1. The proprioception-dropout row of block E compares two policies of the same family that differ only in proprioception dropout with probability 0.1, read at each policy's best checkpoint with an evaluation implementation different from that of the camera-dropout row. Both comparisons use 1,000 episodes per regime.

Augmentation and encoder adaptation. On LIBERO, block F compares encoder LoRA, image augmentation, and both combined against the plain, either-camera-dropout policy of block D; the variants run the encoder online and are selected at 120k updates, whereas the plain reference uses stored features and is selected at 90k. On RoboTwin, the frozen-encoder policy without augmentation is compared with the augmentation-and-LoRA policy trained with the final augmentation pipeline, both at 150k updates and evaluated on 5,000 episodes per regime; the plain policy uses 2,400 of the 2,500 demonstrations and the augmentation-and-LoRA policy all 2,500, and the evaluation implementations also differ. Because the policies also differ in training duration on LIBERO and in training data on RoboTwin, block F measures recipe-level gains rather than isolated effects.

B Training and Evaluation Protocols

This appendix specifies the headline training recipe, the data and checkpoint selection behind the reported La-JEPA results, the benchmark protocols, and the sources of baseline results.

Table 7: Architecture, training, and inference settings.

Policy size	S	B	L
Width / layers / heads	384 / 12 / 6	768 / 12 / 12	1024 / 24 / 16
LoRA blocks (LIBERO)	Last 8	Last 8	All 24
LoRA blocks (RoboTwin)	Last 8	Last 8	Last 8
Transformer / losses	Equal World/Action DiT widths and depths; MLP expansion 4; action and future loss weights 1 each		
Optimizer	AdamW; policy LR 5×10^{-4}; betas $(0.9, 0.95)$; matrix weight decay 0.01; gradient clipping 1		
Precision / EMA	BF16 mixed precision; EMA decay 0.9999		
Schedule	150,000 updates; 5% warmup, cosine decay, final 10% constant tail; LR floor 1% of peak		
Global batch	LIBERO: 128; RoboTwin: 400		
LoRA	Rank 16; $\alpha = 32$; dropout 0.05; LR 10^{-4} with cosine decay to 5% of peak and 1% in the final tail; weight decay 0		
Adapter locations	Fused Q/K/V, attention output, both MLP projections		
Conditioning dropout	Text and proprioception: 0.1 each; camera: 0.1, either view on LIBERO, wrists only on RoboTwin; retain at least one view		
Action/state normalization	Per-channel min/max on LIBERO; z-score on RoboTwin		
Feature normalization	Training-set statistics; separate motion-target statistics; LIBERO pooled futures use per-camera scalar variance correction from 64 training windows		
Inference defaults	$\tau_{\text{ctx}} = 0$; predict 32 actions; execute 10 per replan on LIBERO, 24 on RoboTwin		

Table 8: Observation augmentation. Probabilities are conditional on not bypassing augmentation; fog is one of three corruption types.

Operation	Probability	Parameters and scope
Unmodified window	0.3	Bypass all observation augmentation
Photometric	0.6	All cameras share brightness/contrast/saturation strength 0.3, hue 0.03 turns, gamma $[0.8, 1.25]$
Corruption	0.5	Primary camera; uniform choice of Gaussian blur ($\sigma \in [0.5, 6]$ pixels), motion blur (length $[5, 20]$ pixels), or fog
Fog	Within corruption	Opacity $[0.2, 0.6]$; contrast multiplier $[0.5, 0.9]$; grid size 4–8; gray fraction 0.5–1.0
Rotation	0.5	Primary camera; yaw/pitch $[-4, 4]^\circ$, roll $[-3, 3]^\circ$; homography KRK^{-1}, replicated borders, consistent camera-pose update; vertical FOV 45° (LIBERO), 37° (RoboTwin)

Architecture and optimization. Table 7 summarizes the architecture, optimization, conditioning, and inference settings of the headline recipe at the three policy sizes.

Observation augmentation. Table 8 lists the final augmentation recipe. After the gate that leaves a window unmodified, transformation gates are independent, and parameters are shared across frames. Only observations are augmented; future targets are computed from unmodified frames by the frozen encoder.

Precomputed features and token reduction. Before training, we encode every demonstration once with the frozen V-JEPA 2.1 encoder and store the resulting features. Stored-feature policies, used in most design studies, read both observations and future targets from these stores and never run the encoder during training. Encoder LoRA and image augmentation require encoding the observations at every training step, so the final policies encode their observation clips online but still read their future targets from the stores. Encoding the targets online as well would dominate the encoder cost of each step, since each camera's targets span 16 frames, eight times as many encoder

tokens as its two-frame observation clip. Targets from the adapting encoder also do not improve success (Appendix E.1), so the targets remain those of the frozen encoder.

The features are then reduced spatially before entering the World DiT. Mean pooling, the simplest reduction, averages each 2×2 neighborhood of a feature grid into one token. Because it can be applied once when the features are precomputed, it shrinks the stored features fourfold. Most of our development therefore used pre-pooled stores, which are much smaller and far easier to load during training than full-resolution V-JEPA features. Patch grouping instead concatenates neighboring features into one wider token, reducing the number of tokens without discarding any feature values, but the stored features keep their full size. Mean pooling suffices on LIBERO, whereas on RoboTwin the detail it averages away matters: patch grouping raises randomized success by 4.1 points (Appendix A.1), so the final RoboTwin recipe uses patch grouping.

Training data and checkpoint selection. On LIBERO, S trains on the 95% split (1,663 of 1,743 episodes; Appendix A), whereas B and L use all 1,743 episodes. All LIBERO headline checkpoints are selected at 120,000 updates. Because S uses the 95% split, S-to-B/L comparisons also change data coverage, and L additionally uses full-depth LoRA. The final RoboTwin S, B, and L policies train on all 2,500 clean demonstrations and are selected on a 1,000-episode clean grid disjoint from the evaluation episodes, which picks 150k updates for S and 120k for B and L; they are then evaluated on 5,000 episodes per condition with action noise redrawn at every replan.

Benchmark protocols. LIBERO scores are success rates over the four standard suites. LIBERO-Plus scores are success rates over all 10,030 evaluation episodes of the seven perturbation categories, including background textures, weighting every episode rather than every category equally. Literature aggregates are recomputed from reported per-category rates when needed. La-JEPA's headline LIBERO-Plus evaluations use the upstream suite-specific step limits (220, 280, 300, and 520 steps, after 10 settling steps). Its headline LIBERO results apply the same per-suite limits to rollouts recorded with a 700-step budget, counting as failures the successes reached after a suite's limit; these rollouts use no settling steps. The design studies, evaluation-setting sweeps, and compiled-build validation keep a flat budget in every suite, 700 steps except for the LIBERO pair of block B (600 steps after 30 settling steps; Appendix A.1), so each comparison is made within one protocol. RoboTwin policies are trained on clean demonstrations and evaluated in clean and randomized scenes, and the mean of the two regimes is the leaderboard aggregate. La-JEPA's RoboTwin policies are conditioned on one fixed instruction per task, the task's full description, in both training and evaluation. JEPA-WAM reports 20 RoboTwin tasks, so Table 2 restricts our evaluations to the same tasks, using 2,000 episodes per condition. JEPA-WAM trains only on these 20 tasks, whereas our policies train on all 50. The subset is easier than the full benchmark: per-task DP3 and ACT policies reach 73.9% and 51.8% clean success on it (Lin et al., 2026), against 55.2% and 29.7% over all 50 tasks (Chen et al., 2025). Projected to all 50 tasks, JEPA-WAM's success is estimated at 64.1% clean and 26.9% randomized (see the full-suite estimate below), below every La-JEPA size in Table 1. Methods pretrained on robot trajectories beyond the target demonstrations are outside the main comparison and are explicitly identified in the latent world-action model comparison, the leaderboard figure, and the physical-robot comparison.

Baseline sources. Qwen-RobotManip and StarVLA (StarVLA Community, 2026) results, and Cosmos Policy's LIBERO-Plus score, come from the Qwen-RobotManip report (Yuan et al., 2026a); the remaining LIBERO and LIBERO-Plus scores come from each method's paper or the reproductions cited below. Where a paper prints a category-averaged LIBERO-Plus score (Faster-WAM, ImageWAM, JEPA-WAM, LeapBot-WA, and MINT), we recompute the episode-weighted score from its per-category rates, so our values can differ from the printed averages. Fast-WAM's clean-trained RoboTwin scores come from the official RoboTwin 2.0 leaderboard[1] (Chen et al., 2025), and its LIBERO-Plus result from the Robust-WAM reproduction (Yan et al., 2026). Faster-WAM reports RoboTwin results only for training on clean and randomized demonstrations, so we train it on the clean demonstrations ourselves, adapting the open-source Fast-WAM code as described in the Faster-WAM paper. LeapBot-WA's clean-plus-randomized training regime is excluded from the RoboTwin comparison of Table 2. In Figure 6, SmolVLA's LIBERO scores come from its variants without robot pretraining (Shukor et al., 2025); and the ACT and DP3 RoboTwin scores

[1] https://robotwin-platform.github.io/leaderboard

Table 9: Policy-call latency (ms): batch size one, median of 100 calls after 20 warm-ups on one consumer GPU. OOM: out of memory; dashes: not measured. The timed L uses last-eight-block rather than full-depth LoRA, which does not change latency.

Policy	Steps	Optimized	Original-kernel CUDA graphs	Eager
La-JEPA S	10	17.78	34.20	99.96
La-JEPA B	10	25.16	42.35	98.21
La-JEPA L	10	59.72	88.87	167.93
Fast-WAM	10	88.09	OOM	193.44
Faster-WAM	10	50.96	OOM	62.70
$\pi_{0.5}$	10	71.89	—	—
La-JEPA S	3	11.04	19.59	—
La-JEPA B	3	13.40	22.54	—
La-JEPA L	3	26.30	39.37	—

come from the RoboTwin 2.0 benchmark report (Chen et al., 2025). Figure 4 additionally includes embodied-pretrained policies from the official RoboTwin leaderboard (4D-WAM (Yang et al., 2026), GigaBrain-0.7 (GigaBrain Team et al., 2026), and OLA-Sem and OLA-Geo, which are leaderboard entries without an accompanying paper), as well as the embodied-pretrained Qwen-RobotManip model from its clean-to-randomized comparison (Yuan et al., 2026a). In that figure, the La-JEPA point is B, and the dashed contour fixes the arithmetic mean of clean and randomized success.

Action-noise scale for L. All headline results use the default action-noise scale of 1, which multiplies the initial action noise at each replan; for L, this gives 82.26% on LIBERO-Plus. Appendix E.2 reports a sweep over scales $\{0, 0.25, 0.5, 1\}$, in which scale 0.5 reaches 82.50%; no scale gives a statistically supported improvement, so we report the default. The scale does not change the trained model or its parameter count.

JEPA-WAM full-suite estimate. For the asterisk-marked points in Figures 1 and 4, we fit separate linear regressions from 20-task to 50-task RoboTwin success using seven policies with both scores: Diffusion Policy (Chi et al., 2023), ACT (Zhao et al., 2023), RDT (Liu et al., 2024), π_0 (Black et al., 2024), and DP3 (Ze et al., 2024) from the official leaderboard's clean-trained per-task results, and La-JEPA S and B. With success expressed in percent, the fits are $\hat{s}_{\text{clean}} = -29.83 + 1.176 s_{20}$ and $\hat{s}_{\text{rand}} = -1.88 + 0.780 s_{20}$, with leave-one-out mean absolute errors of 2.1 and 1.4 points, respectively. Applied to JEPA-WAM's reported 79.9% clean and 36.9% randomized success (Lin et al., 2026), they give estimates of 64.1% and 26.9% (mean 45.5%). Figure 1 shows the mean estimate at 1.1 B deployed parameters; these cross-policy estimates are excluded from the result tables.

C Inference Cost and Model Size

This appendix defines how we time policy calls and count parameters, the two cost measures of Section 5.3.

C.1 Latency

Protocol. Tables 9 and 3 time complete benchmark policy calls, not executed actions or control-loop cycles. Each timed call includes fresh instruction tokenization and encoding, input preparation, transfer, and host-side action output; loading, compilation, simulation, and transport are excluded. Text and vision encoders run on parallel streams where supported. Eager execution is uncompiled; original-kernel CUDA graphs reduce launch overhead without changing kernels; optimized builds also compile the computation. The head-to-head WAM rows use ten denoising steps and 32 predicted actions, whereas the three-step rows and $\pi_{0.5}$ (official JAX JIT, ten steps, native ten-action horizon) are separate operating points. Each method keeps its native inputs: La-JEPA uses two 256×256 cameras with two adjacent frames and FLAN-T5-large, timed in FP32 (our success

evaluations use FP16), whereas in our setup Fast-WAM and Faster-WAM use two 224×224 single-frame cameras and BF16 UMT5-XXL (Chung et al., 2023). Action noise is pre-drawn and copied inside the timed call for La-JEPA, and drawn on the CPU inside the call for Fast-WAM. LingBot-VA is timed on the same device under the same protocol with its native implementation and appears, rounded to 10 ms, only in Table 3.

Execution variants. For La-JEPA, original-kernel CUDA graphs reproduce eager actions bit-for-bit across 36 replays per model on 12 inputs that vary observations, instructions, and noise. Compilation is not bit-exact: the maximum ten-step action differences are 0.010, 0.021, and 0.009 for S, B, and L. Our simulation success rates use uncompiled execution, which the CUDA graphs reproduce exactly, whereas the physical-robot La-JEPA policy runs compiled. For Fast-WAM and Faster-WAM, original-kernel graphs exceed the available GPU memory in our implementation.

Closed-loop validation. We evaluate the exact timed builds on 2,000 LIBERO episodes with a 700-step budget in every suite (Appendix B), with action noise of scale 1 redrawn at every replan, and compare each with uncompiled execution of the same checkpoint using paired McNemar tests. Compiled ten-step builds reach 95.20%, 97.05%, and 96.25% for S, B, and L, against 94.30%, 97.35%, and 96.50% uncompiled ($p = 0.15$, 0.59, and 0.69). In separate evaluation runs, ten-step CUDA-graph builds reach 94.70%, 97.00%, and 96.35% ($p = 0.55$, 0.50, and 0.84), and the three-step B graph build 96.90% against 97.30% ($p = 0.45$). None of the seven paired comparisons differs significantly. The timed L build applies LoRA to the final eight encoder blocks and is selected at 150k updates.

Serial encoding. With serial rather than parallel text and vision encoding, optimized ten-step calls take 19, 26, and 61 ms for S, B, and L, against 92 ms for Fast-WAM and 54 ms for Faster-WAM, leaving the speedups essentially unchanged.

C.2 Parameter Accounting

Definitions. Table 10 lists the components behind the size columns. Trainable counts the weights updated on target demonstrations, including LoRA adapters. Deployed counts the policy and encoder parameter inventory, including frozen visual and language encoders; it is a size measure, not a measurement of memory use or operations per action. Deployed counts merge LoRA adapters into the encoder weights, adding no parameters, and count shared encoders once across cameras. Baseline counts use measured checkpoints or published inventories; where encoder-only counts are unavailable, we count full tokenizer inventories, including unused decoder weights, which conservatively overestimates executed parameters. Transformer initialization refers to the trained world-action predictor or VLA backbone: video or image generation, vision-language pretraining, or random initialization.

RoboTwin configurations. Table 10 and the main-text size columns refer to LIBERO configurations. RoboTwin's token projections change La-JEPA's sizes to 0.11/0.75 B (trainable/deployed) for S, 0.37/1.0 B for B, and 1.2/1.8 B for L. Figure 6 complements Figure 1 by plotting success against trainable rather than deployed parameters.

D Real-Robot Setup

This appendix describes the physical platform, the four tasks, demonstration collection, and the training and evaluation protocol behind Table 3, and reports completion times.

Robot, cameras, and control. A tabletop Franka Panda with a parallel gripper uses a wrist-mounted RealSense D405 camera and a fixed external RealSense D435 camera. A GPU workstation runs Linux with a real-time kernel. Policies receive both camera views, proprioception, and a task instruction, and execute end-effector deltas and gripper commands through Polymetis[2] and the Franka Control Interface.

[2] https://facebookresearch.github.io/fairo/polymetis/

Table 10: Parameter inventory (billions); La-JEPA rows for LIBERO. m: measured, r: reported, e: estimated.

Method	Trained components	Frozen components at inference	Trainable	Deployed
Qwen-RobotManip-scratch	Qwen3.5-4B, action head	—	4.7^{e}	4.7^{e}
StarVLA	Qwen3-VL-4B, action head	—	4.6^{e}	4.6^{e}
Fast-WAM	Wan2.2 video DiT (5.00), action DiT (1.02)	UMT5-XXL encoder (5.68), Wan2.2 VAE (0.71)	6.02^{m}	12.4^{m}
Faster-WAM	Wan2.2 video DiT (5.00), action head (0.05)	UMT5-XXL encoder (5.68), Wan2.2 VAE (0.71)	5.05^{r}	11.4^{e}
Cosmos Policy	Cosmos-Predict2-2B DiT (1.96)	T5-XXL encoder (4.86), Wan2.1 VAE (0.13)	2.0^{r}	7.0^{e}
ImageWAM	FLUX.2 4B image DiT (3.88), action DiT (0.64)	Qwen3-4B encoder (4.02), VAE (0.08)	4.5^{e}	8.6^{e}
JEPA-WAM	Qwen2.5-0.5B LoRA (0.004), action DiT (≈0.30–0.37), transition head (0.004)	Qwen2.5-0.5B (0.49), V-JEPA 2.1 L (0.30), projector (0.002)	0.3^{e}	1.1^{e}
$\pi_{0.5}$	full model	—	3.6^{e}	3.6^{e}
La-JEPA S	world and action transformers (0.094), encoder LoRA (0.002)	V-JEPA 2.1 L (0.305), FLAN-T5-large encoder (0.341)	0.096^{m}	0.74^{m}
La-JEPA B	world and action transformers (0.335), encoder LoRA (0.002)	V-JEPA 2.1 L (0.305), FLAN-T5-large encoder (0.341)	0.337^{m}	0.98^{m}
La-JEPA L	world and action transformers (1.131), full-depth encoder LoRA (0.006)	V-JEPA 2.1 L (0.305), FLAN-T5-large encoder (0.341)	1.14^{m}	1.78^{m}

Tasks and objects. Cube placement uses a red wooden cube and a bowl. Fruit picking requires removing a bamboo steamer's lid, setting it down, and then picking the correct fruit, a plum, from inside the steamer. Push T requires pushing a wooden T onto a T shape marked on the table. Dish manipulation requires placing both plates, then the spoon on the gray plate and the fork on the pink plate, in one uninterrupted rollout; plate locations are fixed, but the utensils' starting positions are interchangeable. Every training demonstration and evaluation rollout contains the complete uninterrupted sequence of its task. Figure 5 shows the initial scene of each task from the external camera.

Demonstration collection. SpaceMouse teleoperation provides six-degree-of-freedom end-effector control through the same interface. We record camera observations, robot state, end-effector pose, and gripper state at 10 Hz, collecting 100 demonstrations per task (400 in total). Initial cube positions vary within $10\,\mathrm{cm} \times 10\,\mathrm{cm}$, and plum positions vary within the steamer; Push T and dish manipulation use smaller placement variations, including utensil rearrangements.

Training and evaluation. Policies train jointly on the four language-conditioned tasks using their respective recipes. Fast-WAM and Faster-WAM train for 30 epochs; La-JEPA trains its B policy transformer from scratch with encoder LoRA but without image augmentation. Table 3 reports counts over 20 rollouts per task, with explicit ten-rollout denominators for LingBot-VA's dish result. Utensil 1 and Utensil 2 denote successful placement of the first and second utensils on plates, irrespective of utensil identity or order. Position additionally requires the correct spoon–gray and fork–pink assignments and success at all preceding stages; the two plate stages share one table column because their counts are identical. Average weights the four tasks equally, using Position for

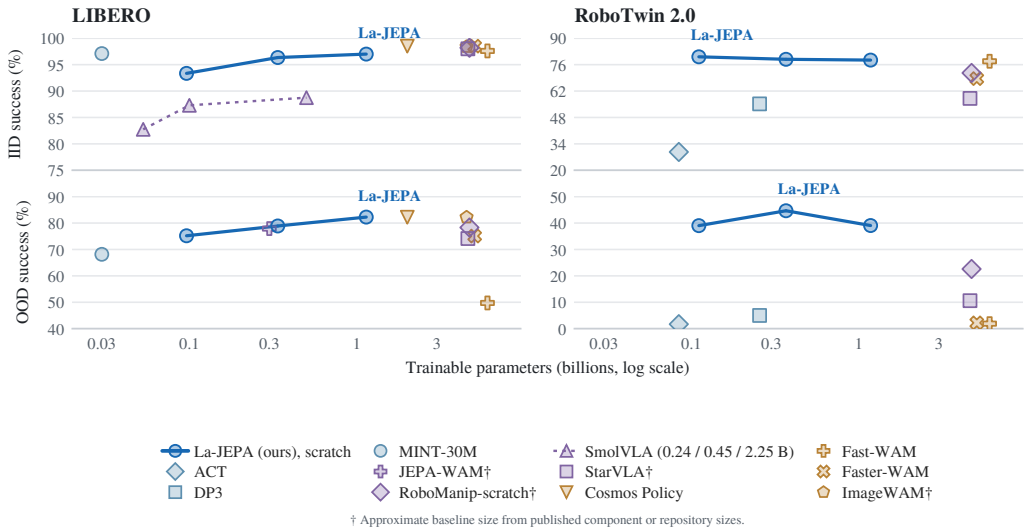


Figure 6: Companion to Figure 1: in-distribution (top) and out-of-distribution (bottom) success against trainable parameters on a logarithmic axis, for LIBERO (left) and clean-trained RoboTwin 2.0 (right). Trainable counts every weight updated on the target demonstrations, adapters included. La-JEPA points are blue and connected in size order; purple denotes VLM-initialized and brown generator-initialized action transformers, while other scratch policies use muted blue. SmolVLA (Shukor et al., 2025) is shown at its three sizes joined by a dotted line.

dish, in mean successes out of 20; LingBot-VA's average is omitted because its dish result uses 10 trials rather than 20.

Completion time. Table 11 reports the mean and standard deviation of completion time over successful trials. La-JEPA is the fastest policy without embodied pretraining on every task and the fastest overall on fruit picking and Push T, while $\pi_{0.5}$ is fastest on cube placement and dish manipulation. Our main hypothesis for LingBot-VA's low Push T and dish success is that it requires substantially more real-robot finetuning than the other policies; independently of success, its completion times, three to eleven times longer than those of the other policies, make it impractical for closed-loop control on this platform. Its dish entry comes from its single successful trial.

Table 11: Real-robot completion time over successful trials (s; standard deviation in small type). Bold: fastest; underlined: second. †Single successful trial.

	With embodied pretraining		Without embodied pretraining		
Task	$\pi_{0.5}$	LingBot-VA	Fast-WAM	Faster-WAM	La-JEPA
Fruit	<u>24.4</u> ± 2.7	175.3 ± 41.0	36.7 ± 9.0	35.1 ± 9.5	**21.4** ± 0.8
Cube	**10.8** ± 0.5	117.8 ± 50.9	21.9 ± 7.6	18.2 ± 4.6	<u>11.5</u> ± 3.2
Push T	<u>22.8</u> ± 14.7	112.5 ± 9.1	24.6 ± 6.6	23.8 ± 12.1	**21.1** ± 4.1
Dish	**39.0** ± 3.4	269.0†	85.0 ± 15.3	72.0 ± 8.8	<u>58.1</u> ± 8.7

E SUPPLEMENTARY RESULTS

This appendix collects results that inform but do not define the final recipe: adapted future targets, evaluation-time settings, future-only pooling, representation diagnostics, and per-task RoboTwin success.

Table 12: Per-task RoboTwin 2.0 success of La-JEPA S, B, and L (%), 100 episodes per task and condition.

	S		B		L	
Task	Clean	Rand.	Clean	Rand.	Clean	Rand.
Adjust bottle	70	15	63	22	71	10
Beat block hammer	88	17	90	21	88	23
Blocks ranking rgb	95	34	94	45	93	58
Blocks ranking size	55	23	66	35	60	37
Click alarmclock	100	46	100	63	100	62
Click bell	100	47	100	64	100	59
Dump bin bigbin	98	79	98	76	96	76
Grab roller	100	69	97	72	100	67
Handover block	90	6	78	7	89	5
Handover mic	80	27	94	31	70	33
Hanging mug	55	15	46	11	46	14
Lift pot	98	13	99	10	100	12
Move can pot	72	5	59	6	62	7
Move pillbottle pad	72	11	82	24	62	10
Move playingcard away	90	46	90	49	71	26
Move stapler pad	39	7	35	9	29	2
Open laptop	93	13	72	10	85	8
Open microwave	85	38	96	20	95	11
Pick diverse bottles	80	31	78	29	77	33
Pick dual bottles	96	36	86	26	91	32
Place a2b left	75	30	57	34	63	21
Place a2b right	56	21	52	29	57	20
Place bread basket	90	70	93	84	92	81
Place bread skillet	92	42	90	45	92	36
Place burger fries	97	74	95	81	98	71
Place can basket	5	8	9	25	1	14
Place cans plasticbox	100	60	98	93	98	73
Place container plate	99	80	99	92	98	83
Place dual shoes	64	41	67	49	69	37
Place empty cup	98	58	100	73	98	51
Place fan	70	7	54	18	50	7
Place mouse pad	51	13	45	16	51	7
Place object basket	87	58	88	45	89	40
Place object scale	76	18	76	25	79	20
Place object stand	94	63	93	67	93	86
Place phone stand	85	68	85	66	80	31
Place shoe	85	65	86	73	91	66
Press stapler	88	27	84	46	84	30
Put bottles dustbin	84	66	81	63	84	74
Put object cabinet	47	34	49	36	50	30
Rotate qrcode	73	13	53	57	69	11
Scan object	62	53	81	46	84	60
Shake bottle	100	48	100	63	100	73
Shake bottle horizontally	100	48	100	64	100	74
Stack blocks three	99	57	99	34	98	27
Stack blocks two	100	56	100	73	100	67
Stack bowls three	87	68	85	73	82	70
Stack bowls two	93	80	99	92	91	76
Stamp seal	38	8	46	11	36	9
Turn switch	57	39	55	30	59	23
Mean over 50 tasks	80.2	39.0	78.8	44.7	78.4	39.1

Table 13: Evaluation-setting sweeps at fixed checkpoints (success in %). LIBERO values use the 700-step budget in every suite (Appendix B); B's LIBERO reference is a re-evaluation of its default setting.

LIBERO

Model	Evaluation setting	Value	Success (%)
S	Reference	Default	94.30
S	Denoising steps	3	94.50
L	Reference	Default	97.40
B	Reference	Default	97.60
L	Execution chunk	5	97.00
L	Execution chunk	8	96.30
B	Execution chunk	5	96.40
B	Execution chunk	8	97.30
L	Denoising steps	3	96.45
L	Denoising steps	5	97.05
L	Denoising steps	20	96.95
L	Denoising steps	50	96.60
B	Denoising steps	3	97.30
B	Denoising steps	5	97.25
B	Denoising steps	20	96.80
B	Denoising steps	50	96.80
L	Noise scale σ	0.25	96.65

LIBERO-Plus

Model	Evaluation setting	Value	Success (%)
L	Reference	Default	82.26
B	Reference	Default	78.89
L	Execution chunk	5	78.97
L	Execution chunk	6	80.19
L	Execution chunk	8	81.68
L	Execution chunk	16	81.53
L	Execution chunk	24	78.46
L	Execution chunk	32	74.92
B	Execution chunk	5	75.44
B	Execution chunk	8	78.46
L	Denoising steps	3	82.41
L	Denoising steps	5	82.12
L	Denoising steps	20	81.88
L	Denoising steps	50	81.66
B	Denoising steps	3	79.22
L	Noise scale σ	0	82.44
L	Noise scale σ	0.25	82.24
L	Noise scale σ	0.5	82.50
B	Noise scale σ	0.25	79.38

E.1 Adapting Future Targets

On LIBERO with L, we replace the frozen future targets with targets from an EMA of the adapting encoder, keeping the clips, observation processing, and normalization statistics of the frozen-target policy and passing no gradients through the target path. The adapted-target policy, selected at 90k updates, achieves 81.65% on LIBERO-Plus, against 82.26% for the frozen-target policy at 120k, both at action-noise scale 1. Adapted targets do not improve success (paired McNemar $p = 0.096$), so the final recipe retains frozen targets.

E.2 Evaluation Settings

At the selected 120k checkpoints, we vary the execution chunk length, the number of denoising steps, and the action-noise scale for L and B, and test three-step inference for S (Table 13); the reference executes ten actions per replan with ten denoising steps and noise scale 1, and LIBERO runs use a 700-step budget in every suite (Appendix B). Executing only five actions reduces LIBERO-Plus success by 3.29 points for L and 3.45 for B, whereas the number of denoising steps matters less: with three steps, B and L stay within one point of their ten-step references on both benchmarks, and S obtains 94.50% against 94.30% on LIBERO ($p = 0.81$, paired McNemar). Three-step calls take 23 ms for B and 39 ms for L with original-kernel CUDA graphs (Table 9), an operating point separate from the ten-step latency comparison. No tested noise scale gives a statistically supported aggregate improvement in the paired comparisons; the headline L result uses the default scale 1 (82.26%; Appendix B), and the best single scale, 0.5, reaches 82.50%.

E.3 Future-Only Pooling

Complementing the token-reduction comparison (Appendix A.1), we keep patch-grouped observations and mean-pool only the future targets (Table 14). In-distribution success stays near 61–62%, whereas out-of-distribution success is non-monotonic in pooling granularity. Because the settings also differ in positional geometry, target normalization, camera weighting, and checkpoint step, these scores describe the tested recipes rather than an isolated effect of target resolution. The poli-

Table 14: Future-only pooling on RoboTwin (success in %).

Future pooling	IID	OOD
$2\times$ head, $4\times$ wrists	61.24	15.68
$4\times$ head, $8\times$ wrists	61.96	18.56
Global spatial pooling	61.14	15.04

Table 15: Representation diagnostics with and without future prediction on RoboTwin.

Training objective	Clean success (%)	Effective rank	Action readout R^2
Action + future prediction	66.5	137.70	0.4315
Action only	62.8	48.41	0.7937

cies are small, frozen-encoder, static-feature policies trained on the 95% split. Observations keep patch grouping with spatial factors $[2, 4, 4]$ for the head, left-wrist, and right-wrist cameras, and the future targets are pooled by $[2, 4, 4]$, $[4, 8, 8]$, or global averaging, giving $[80, 20, 20]$, $[20, 6, 6]$, or $[1, 1, 1]$ spatial tokens per temporal position. The first setting has the patch-grouped reference's token count but averages rather than concatenates features. The rows report 150k, 60k, and 90k checkpoints, each evaluated in both regimes on 5,000 episodes, under an evaluation implementation that differs from the other studies.

E.4 Representation Diagnostics

Table 15 analyzes frozen-encoder RoboTwin policies trained with and without future prediction at 150k updates, separate from the policies in Figure 3A. Centered RankMe (Garrido et al., 2022) uses 2,000 sampled, mean-pooled final-layer value features, and a cosine 5-nearest-neighbor action readout reports R^2 over normalized action chunks. Features span 100 episodes held out from policy training, with an episode-disjoint 70/30 split for fitting and testing the readout and with language and proprioceptive conditioning retained. Future prediction raises effective rank together with clean success under the 1,000-episode protocol, but lowers action-readout R^2, so nearest-neighbor action decodability does not explain the difference in control performance.

E.5 Per-Task RoboTwin Results

Table 12 lists the per-task RoboTwin 2.0 success of La-JEPA S, B, and L, the policies reported in Table 1, trained on clean demonstrations and evaluated on 100 episodes per task in clean and randomized scenes. Success varies widely across tasks, and randomized scenes lower success on almost every task.