MINIWAM: LEARNING COMPACT FUTURE TARGETS FOR EFFICIENT WORLD-ACTION MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

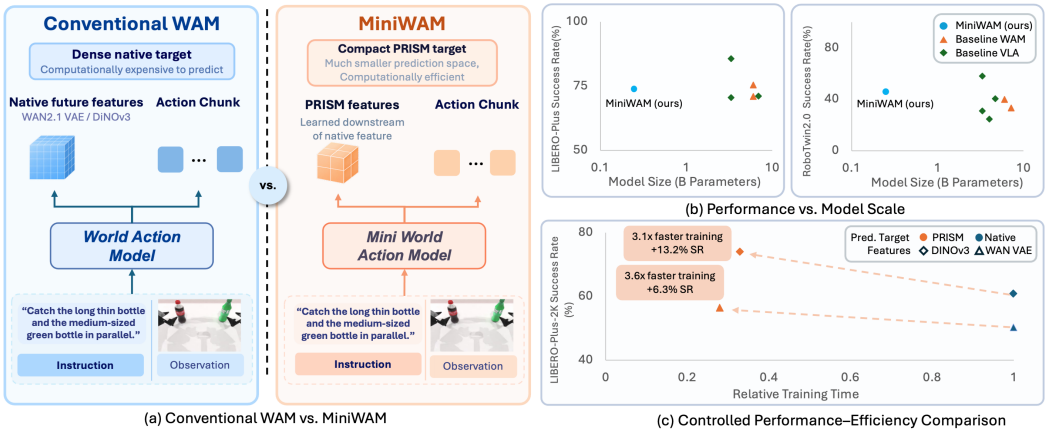
World modeling has emerged as an effective co-training objective for robot policies, giving rise to World Action Models (WAMs) that jointly predict actions and future states. However, most WAMs predict future states in the native representation space of pretrained visual backbones, resulting in high-dimensional targets with substantial training cost. We introduce MiniWAM, which instead predicts compact future representations learned from privileged current–future transitions. To construct these targets, we propose Predictive Representations via Inverse Spatiotemporal Modeling (PRISM), which combines inverse-dynamics supervision with feature reconstruction to emphasize control-relevant transition information while preserving useful future-state information. With the learned PRISM encoder frozen, MiniWAM is trained to jointly predict the resulting targets and robot actions from current observations. With $65\times$ fewer native future feature tokens, MiniWAM consistently outperforms native future-feature prediction with both DINOv3 and WAN2.1 VAE features, while achieving up to an $8\times$ speedup in world–action training. At 0.25B parameters, MiniWAM is already competitive with substantially larger WAMs on LIBERO, LIBERO-Plus, and Robotwin 2.0 simulation benchmarks. Representation analyses further show that PRISM contributes behavioral structure beyond feature reconstruction alone. These results demonstrate that effective world–action modeling does not require predicting native visual futures, and that compact predictive representations provide a strong and substantially more efficient target for policy learning.

1 INTRODUCTION

Learning to predict future observations alongside robot actions provides a powerful training signal for robot control. Recent World Action Models (WAMs) exploit this idea by jointly learning robot actions and future visual states, encouraging policy representations to capture environment dynamics relevant to control (Kim et al., 2026; Ye et al., 2026; Yuan et al., 2026b;a; Cen et al., 2025). This paradigm has led to strong results across simulated and real-world manipulation. However, it also introduces a substantial computational burden. Most WAMs model the future through high-dimensional visual representations, typically dense video-VAE latents inherited from video-generation models (Wan et al., 2025). Predicting these representations requires processing large spatiotemporal token sequences, making world modeling considerably more expensive. Yet preserving every detail in the native feature space may not be necessary for action learning.

Prior work has increasingly sought to improve the efficiency of WAMs by reducing the cost of future generation (Yuan et al., 2026b), sparsifying world–action interaction (Zhao et al., 2026), or shrinking the model and token budget devoted to visual prediction (Li et al., 2026). A complementary line of work replaces dense visual prediction with compact latent future supervision, using representations of future observations to shape policy-internal latent states (Zheng et al., 2025; Su et al., 2026; Luo et al., 2026). Together, these approaches suggest that the benefits of future modeling need not rely on explicit high-dimensional visual rollouts. However, existing efficiency-oriented WAMs still retain rich visual representations as their world-modeling targets, while latent approaches primarily incorporate future information through internal policy states or auxiliary conditioning spaces. This leaves a distinct question: can we directly learn a compact yet control-relevant future representation that replaces dense visual prediction as the world-modeling target of a WAM for efficient training?

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.



(a) Conventional WAM vs. MiniWAM

(b) Performance vs. Model Scale

(c) Controlled Performance–Efficiency Comparison

Figure 1: **MiniWAM replaces costly native future prediction with compact predictive representations.** (a) While conventional WAMs predict high-dimensional native future features, MiniWAM jointly predicts actions and compact PRISM targets learned from the same visual features via inverse dynamics and reconstruction. (b) MiniWAM remains competitive with substantially larger WAMs and VLAs on LIBERO-PLUS. (c) In controlled comparisons across VAE and DINOv3 feature spaces, PRISM simultaneously improves success rate and reduces training time relative to native future-feature prediction.

To learn such a representation, we propose Predictive Representations via Inverse Spatiotemporal Modeling (PRISM). We aim to learn a compact representation that preserves information about both the actions underlying an observed transition and the resulting visual state. During representation learning, a bottleneck encoder compresses current and future visual features into a compact set of latent tokens. Inverse-dynamics supervision encourages these tokens to retain transition information informative of the underlying actions, while feature reconstruction encourages them to preserve future-state content, without requiring the full detail of the native visual representation.

We then introduce **MiniWAM**, which separates learning *what to represent* about the future from learning *how to predict* it. First, PRISM learns to encode current–future visual transitions into compact targets through inverse-dynamics supervision and feature reconstruction. The representation encoder is then frozen, and MiniWAM is trained to jointly predict the resulting PRISM target and robot actions given the current observation. In this way, MiniWAM changes what the world model branch is trained to predict, rather than merely reducing the cost of predicting an otherwise unchanged visual future.

PRISM is not tied to a particular visual feature space, because the bottleneck is learned downstream of the visual representation. We instantiate the framework with both generative WAN2.1 VAE features (Wan et al., 2025) and discriminative DINOv3 features (Siméoni et al., 2025). By replacing dense future representations with a compact set of latent tokens, PRISM reduces the number of future tokens by $65\times$. Despite this substantial compression, MiniWAM consistently outperforms matched variants that directly predict the corresponding native future representations, while requiring substantially less training computation. These controlled comparisons show that the benefit does not arise simply from choosing a stronger visual encoder: replacing its dense future representation with the compact PRISM target improves the performance–efficiency trade-off. This performance–efficiency advantage is illustrated in Figure 1.

Empirically, MiniWAM reduces WAM training cost by up to $8\times$ while outperforming native future-feature prediction across simulation benchmarks. A 0.25B MiniWAM is already competitive with WAMs built upon substantially larger pretrained video-generation backbones. Ablations show that adding inverse-dynamics supervision to reconstruction-only bottlenecks consistently improves performance across bottleneck capacities. These results suggest that strong world–action modeling does not require predicting the native visual future: compact learned representations can provide a stronger and substantially more efficient target for predictive co-training.

2 Related Work

World Models and World–Action Models for Robot Control. World models learn predictive representations of environment dynamics to support planning and policy learning, often in latent or pretrained feature spaces rather than pixel space (Ha & Schmidhuber, 2018; Hafner et al., 2020; Zhou et al., 2025). Building on this principle, recent methods increasingly integrate future prediction directly into policy training, giving rise to World Action Models (WAMs) that jointly model future states and robot actions (Kim et al., 2026; Ye et al., 2026; Yuan et al., 2026b). Using future prediction as a co-training objective, WAMs encourage the policy to capture structured environment dynamics while sharing information between world and action streams. This formulation has achieved strong manipulation performance, but typically requires predicting dense spatiotemporal visual representations, such as video-VAE latents, substantially increasing the cost of policy training.

Efficient and Compact Future Modeling. Recent work improves WAM efficiency through architectural simplification and reduced future-modeling computation (Yuan et al., 2026b; Zhao et al., 2026; Li et al., 2026). A complementary line replaces dense future prediction with compact latent supervision, aligning or supervising policy-internal representations with features derived from future observations (Zheng et al., 2025; Luo et al., 2026; Su et al., 2026). These approaches show that predictive information can be represented more compactly than the original visual future, but typically retain native visual features as world-modeling targets or use compact representations primarily as internal policy states or conditioning signals. Concurrent work (Wang et al., 2026) shows that channel-wise compression of semantic prediction targets can yield strong world–action co-training performance. PRISM instead learns a control-relevant transition representation as the world-prediction target, jointly compressing its spatial and temporal dimensions to substantially reduce the number of predicted tokens and the cost of world–action co-training.

3 Mini World Action Model with Predictive Representations

3.1 Overview

Given a current observation o_t, a future observation window $o_{t+1:t+H_\text{v}}$, and the corresponding action chunk $\mathbf{a}_t = (a_t, \ldots, a_{t+H_\text{a}-1})$, we seek a compact representation of the visual transition that can replace native future features as the predictive target of a World Action Model. Let f denote a frozen visual encoder, and let (x_t, X_t^+) denote the current and future feature blocks obtained by encoding $o_{t:t+H_\text{v}}$ with f. PRISM introduces a spatiotemporal encoder E_ϕ that maps these privileged current–future features to a compact predictive representation $L_t = E_\phi(x_t, X_t^+)$, whose dimensionality is substantially smaller than that of the native future representation X_t^+.

Training comprises two stages, as illustrated in Figure 2. In *Stage 1*, PRISM learns E_ϕ from future transitions using inverse-dynamics supervision and feature reconstruction: the former biases L_t toward control-relevant dynamics, while the latter preserves useful future-state information. In *Stage 2*, E_ϕ is frozen and L_t becomes the world-modeling target of MiniWAM, which jointly predicts the PRISM representation and robot actions from the current observation and conditioning context. Future observations are required only to construct training targets and are unavailable at deployment.

Both stages use a Mixture-of-Transformers (MoT) composed of two modality-specific diffusion transformer (DiT) (Peebles & Xie, 2023) experts: a world expert that models the PRISM representation L_t and an action expert that models $\mathbf{a}_t$. We refer to their tokenized inputs as *world tokens* and *action tokens*, respectively. Each expert retains its own input/output projections, feed-forward layers, and timestep modulation, while mixed self-attention enables information exchange between the two modalities. In Stage 1, the world tokens remain clean and condition action prediction; in Stage 2, the world and action tokens are jointly denoised.

We use flow matching (Lipman et al., 2023) for both prediction branches. For $y \in \{L, \mathbf{a}\}$, we sample Gaussian noise ϵ_y and noise level τ_y and construct

$$\widetilde{y}_{\tau_y} = (1 - \sigma_{\tau_y})y + \sigma_{\tau_y}\epsilon_y, \qquad u_y^\star = \epsilon_y - y.$$

The unified objective is $\mathcal{L}_\text{FM} = \lambda_L \mathbb{E}\big[w(\tau_L)\|\hat{u}_L - u_L^\star\|_2^2\big] + \lambda_a \mathbb{E}\big[w(\tau_a)\|\hat{u}_a - u_a^\star\|_2^2\big]$.

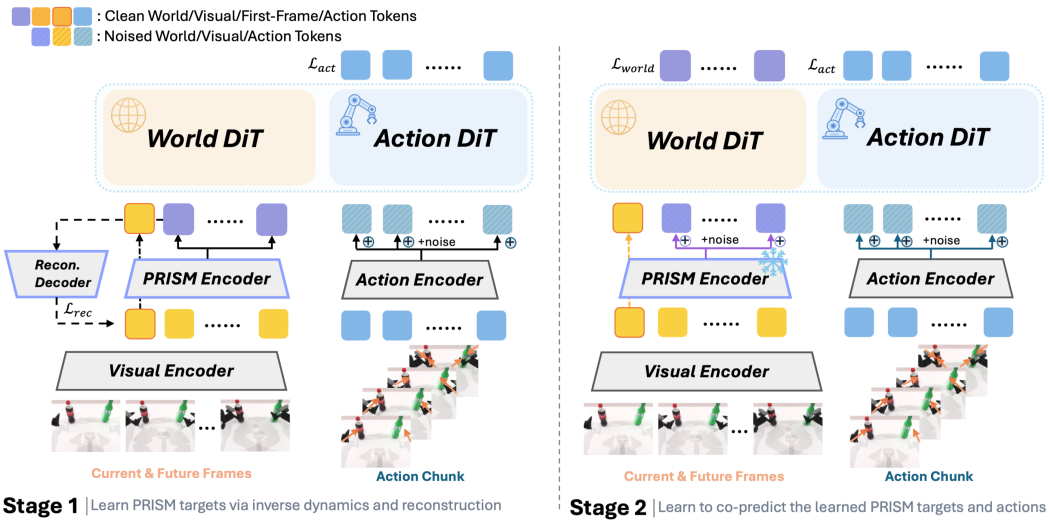


Figure 2: **Two-stage training pipeline of PRISM and MiniWAM.** In **Stage 1**, privileged current–future visual features are compressed by the PRISM encoder into a compact predictive representation L_t, shaped through inverse action prediction and feature reconstruction. The corresponding world tokens remain clean while the action tokens are corrupted for conditional action prediction. In **Stage 2**, the PRISM encoder is frozen and provides compact world-modeling targets; MiniWAM jointly denoises the world and action tokens using its world and action experts. Solid and hatched tokens denote clean and noised tokens, respectively. Language and proprioceptive conditioning are omitted for clarity.

Stage 1 keeps L_t clean and sets $\lambda_L = 0$, using it for inverse action prediction. Stage 2 independently corrupts both L_t and $\mathbf{a}_t$ and optimizes both prediction terms.

3.2 Stage 1: Learning the Predictive Representation

Spatiotemporal transition bottleneck. The transition encoder E_ϕ is instantiated as a lightweight 3D convolutional network. It jointly processes the current and future visual features x_t and X_t^+ arranged along the temporal dimension, progressively compressing them via strided 3D convolutions into a compact spatiotemporal grid $L_t \in \mathbb{R}^{T_L \times H_L \times W_L \times D_L}$, which is subsequently flattened into world tokens for the MoT. We parameterize the bottleneck as a diagonal Gaussian,

$$q_\phi(L_t \mid x_t, X_t^+) = \mathcal{N}\big(\mu_\phi(x_t, X_t^+), \mathrm{diag}(\sigma_\phi^2(x_t, X_t^+))\big), \qquad L_t = \mu_\phi + \sigma_\phi \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

PRISM shapes this bottleneck using two complementary objectives. For inverse dynamics, L_t is provided as a clean conditioning sequence while the action sequence is corrupted according to the flow-matching process above. Direct current-observation and language conditioning are removed from the MoT, forcing visual information relevant to action prediction to pass through L_t. The current feature x_t remains an input to E_ϕ, allowing the bottleneck to represent the current–future transition rather than the future state in isolation.

Feature reconstruction encourages the compact representation to preserve useful information about the native future state. A lightweight decoder D_ψ reconstructs X_t^+ using $\mathcal{L}_{\text{rec}} = \|D_\psi(L_t) - X_t^+\|_2^2$, while a weak KL term regularizes the bottleneck distribution. The Stage-1 objective is

$$\mathcal{L}_{\text{S1}} = \lambda_a \mathbb{E}\left[w(\tau_a)\|\hat{u}_a - u_a^\star\|_2^2\right] + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \beta D_{\text{KL}}\left[q_\phi(L_t \mid x_t, X_t^+) \,\|\, \mathcal{N}(0, I)\right].$$

Inverse dynamics therefore biases the compact state toward behaviorally meaningful transition information, while reconstruction preserves predictive information about the future visual state. The decoder is used only during Stage 1 and is discarded afterward.

3.3 Stage 2: World–Action Modeling with PRISM Targets

After Stage 1, we freeze E_ϕ and use its posterior mean $L_t^\star = \mu_\phi(x_t, X_t^+)$ as a deterministic world-modeling target. MiniWAM receives no future observations as input and learns to predict $L_t^\star$ from the current observation while jointly predicting the action trajectory. The MoT sequence is $[x_t^0, \widetilde{L}_{\tau_L}, \widetilde{\mathbf{a}}_{\tau_a}]$, where x_t^0 denotes clean current-observation tokens, and $\widetilde{L}_{\tau_L}$ and $\widetilde{\mathbf{a}}_{\tau_a}$ denote the noised world and action tokens at noise levels τ_L and τ_a, respectively. Both branches use independent Gaussian noise and, by default, independently sampled noise levels. Language instructions and proprioceptive states are provided as contextual conditioning via cross-attention.

Stage 2 also changes the interaction between the two prediction branches. The world and action tokens attend to one another bidirectionally, allowing world and action predictions to be refined jointly. Both terms of the unified objective are now active:

$$\mathcal{L}_{\mathrm{S2}} = \lambda_L \mathbb{E}\left[w(\tau_L)\|\hat{u}_L - u_L^\star\|_2^2\right] + \lambda_a \mathbb{E}\left[w(\tau_a)\|\hat{u}_a - u_a^\star\|_2^2\right].$$

Thus, MiniWAM retains the standard world–action co-training formulation while replacing native future features with the compact PRISM target $L_t^\star$.

Inference. At deployment, E_ϕ and future observations are no longer needed. Given only the current observation, instruction, and proprioception, we initialize $L^{(K)} \sim \mathcal{N}(0, I)$ and $\mathbf{a}^{(K)} \sim \mathcal{N}(0, I)$ and jointly denoise them through the world and action experts. Each MoT forward pass predicts both velocity fields, after which the two variables are updated according to their respective flow schedules. The final action chunk is executed in a receding-horizon manner.

3.4 Visual Representation Instantiations

PRISM operates downstream of the visual encoder and therefore does not assume a particular representation type. We study two distinct feature families. **VAE features.** We use a frozen WAN2.1 video VAE, representative of the generative latent spaces commonly modeled by video-based WAMs. Its current and future features form a spatiotemporal volume that is compressed by E_ϕ. **DINOv3 features.** We use frozen DINOv3 patch features, temporally aggregating the frame-wise spatial features before E_ϕ compresses the complete current–future sequence into the same compact spatiotemporal representation.

These two instantiations keep the PRISM formulation unchanged while varying the underlying visual feature space. This allows us to test whether learning a compact, control-relevant predictive target is beneficial across both generative and discriminative visual representations.

4 Experiments

We evaluate PRISM to answer three key questions: **(Q1)** Can PRISM replace native future-feature prediction while improving the performance–training-efficiency trade-off? **(Q2)** How does MiniWAM compare with state-of-the-art VLAs and WAMs? **(Q3)** What aspects of future dynamics does PRISM encourage its representation to capture?

4.1 Experimental Setup

We evaluate policy success on LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2026), and RoboTwin 2.0 (Chen et al., 2026), covering controlled world-modeling comparisons, robustness under distribution shifts, and multi-task bimanual manipulation. We additionally benchmark training efficiency on LIBERO and RoboTwin 2.0. Detailed training configurations, evaluation protocols, and hyperparameters are provided in the Appendix.

Benchmarks. LIBERO. LIBERO comprises four standard suites—LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long—with 40 language-conditioned manipulation tasks, and serves as our primary benchmark for controlled comparisons of world-modeling targets. **LIBERO-Plus.** LIBERO-Plus extends these tasks with systematic perturbations to

Table 1: **Controlled comparison of world-modeling targets on LIBERO and LIBERO-PLUS-2K.** All variants use the same 0.25B world–action architecture and training protocol within each visual feature family.

Visual features	Co-training target	LIBERO SR (%) ↑	LIBERO-PLUS-2K SR (%) ↑
VAE	w/o Co-Train	62.3	25.1
	w/ Native Co-Train	92.8	50.2
	w/ PRISM	**94.8**	**56.5**
DINOv3	w/o Co-Train	89.0	56.5
	w/ Native Co-Train	93.2	60.9
	w/ PRISM	**97.2**	**74.1**

camera viewpoints, robot initial states, language instructions, lighting, backgrounds, observation noise, and object layouts, allowing us to evaluate robustness under controlled distribution shifts. **ROBOTWIN 2.0.** ROBOTWIN 2.0 is a 50-task bimanual manipulation benchmark with the Aloha-AgileX robot; we use its multi-task setting and evaluate policy success under both clean and randomized conditions. We additionally use ROBOTWIN 2.0 for training-efficiency measurements, where its additional wrist view increases the visual prediction workload relative to LIBERO.

4.2 PRISM VERSUS DIRECT FUTURE-FEATURE PREDICTION

We first study whether the learned PRISM representation can replace native future features as the co-training target of a WAM. We construct controlled variants for both visual representation families considered in this work. For **Native Co-Train**, the world expert directly predicts the native future representation produced by the corresponding visual encoder: future VAE latent blocks for the VAE variant and the complete sequence of future DINOv3 patch features for the DINOv3 variant. For **PRISM**, the native future features are further compressed by the learned transition encoder, and the resulting compact representation serves as the world-modeling target. We additionally include a **w/o Co-Train** variant for each visual feature family, where the world-prediction objective is disabled. This baseline isolates the contribution of predictive co-training from that of the underlying visual representation. For a controlled comparison, the world expert is adapted only to the dimensionality and tokenization of each target, while the training data, Stage-2 optimization schedule, and number of policy-training updates are kept fixed. PRISM's Stage 1 learns the representation used as its prediction target. All variants are trained on LIBERO and evaluated under matched protocols on LIBERO and LIBERO-PLUS-2K.

Table 1 yields two observations. First, native future-feature co-training substantially improves control across both visual representation families: it raises the VAE-based policy from 62.3% to 92.8% and the DINOv3-based policy from 89.0% to 93.2%. Second, replacing native future features with the learned PRISM representation further improves performance, reaching 94.8% with VAE features and 97.2% with DINOv3 features. Under matched Stage-2 training, these gains demonstrate the benefit of the learned PRISM target. On LIBERO-PLUS-2K, PRISM also improves over native co-training by 6.3 and 13.2 percentage points for VAE and DINOv3, respectively. Notably, PRISM improves over native co-training for both generative VAE features and discriminative DINOv3 features, demonstrating that its benefit is not tied to a particular form of visual representation.

Training efficiency. The performance gain above is accompanied by a substantial reduction in world-modeling computation, as reported in Table 2. Direct prediction requires the world expert to operate on dense native future representations, whereas PRISM compresses the current–future transition into a compact spatiotemporal target, reducing the world-token burden. We benchmark matched PRISM and direct-prediction variants at future horizons of 16 and 32 frames on both LIBERO and ROBOTWIN 2.0 using the same global batch size and hardware. ROBOTWIN 2.0 additionally includes a second wrist camera view, providing a heavier multi-view prediction workload.

PRISM consistently reduces predictive co-training cost across both benchmarks and visual feature spaces, with larger gains under heavier prediction workloads. At horizon 32, estimated total training speedup reaches $8.46\times$ on ROBOTWIN 2.0, while Stage-2 training alone reaches $10.48\times$. Together

Table 2: **Training efficiency of PRISM versus native co-training.** Throughput is measured in samples/s with global batch 256 under matched hardware. S2 is Stage-2 speedup; Total estimates training speedup including Stage 1, assuming equal per-update costs for the two PRISM stages (Appendix D).

Visual Feature	Target	Horizon 16			Horizon 32		
		Throughput ↑	S2 ↑	Total ↑	Throughput ↑	S2 ↑	Total ↑
LIBERO							
WAN2.1 VAE	Native	30.81	1.00×	1.00×	14.20	1.00×	1.00×
	PRISM	**142.04**	**4.61**×	**3.55**×	**124.92**	**8.80**×	**6.77**×
DINOv3	Native	36.17	1.00×	1.00×	15.55	1.00×	1.00×
	PRISM	**143.25**	**3.96**×	**3.05**×	**126.09**	**8.11**×	**6.24**×
ROBOTWIN *2.0*							
WAN2.1 VAE	Native	17.70	1.00×	1.00×	7.79	1.00×	1.00×
	PRISM	**93.77**	**5.30**×	**4.27**×	**81.66**	**10.48**×	**8.46**×
DINOv3	Native	21.48	1.00×	1.00×	8.61	1.00×	1.00×
	PRISM	**93.57**	**4.36**×	**3.51**×	**81.47**	**9.46**×	**7.63**×

Table 3: **Comparison with state-of-the-art robot policies.** † denotes externally reported results. Scale is the reported model size; Embodied PT and Backbone Init. denote embodied pretraining and backbone initialization. "–" means no pretrained backbone and "-" means unavailable.

Type	Method	Scale	Embodied PT.	Backbone Init.	LIBERO SR (%) ↑	LIBERO-PLUS SR (%) ↑	ROBOTWIN 2.0 SR (%) ↑
VLA	OpenVLA-OFT	7B	✓	VLM	96.6	71.4	-
	π_0	3.3B	✓	VLM	93.5	70.5	31.4
	π_0-FAST†	3.3B	✓	VLM	85.5	61.6	-
	UniVLA†	7B	✓	VLM	95.2	42.9	-
	StarVLA-OFT†	4B	×	VLM	96.6	-	24.8
	Xiaomi-Robotics-0†	4.7B	✓	VLM	98.7	-	40.5
	$\pi_{0.5}$	3.3B	✓	VLM	96.4	85.7	58.4
WAM	WorldVLA†	7B	×	VLM	79.1	25.0	-
	Fast-WAM	6B	×	VGM	95.8	71.0	39.9
	DreamWAM	6B	×	VGM	96.9	75.4	-
	AHA-WAM†	7.2B	×	VGM	-	-	33.8
	MiniWAM (ours)	0.25B	×	–	97.2	73.6	46.1

with Table 1, these results show that learned PRISM targets improve the performance–efficiency trade-off of world–action co-training.

4.3 COMPARISON WITH STATE-OF-THE-ART ROBOT POLICIES

We compare MiniWAM with state-of-the-art VLAs (Kim et al., 2025; Black et al., 2025b; Pertsch et al., 2025; Black et al., 2025a; Bu et al., 2025; Community, 2026; Cai et al., 2026b) and WAMs (Cen et al., 2025; Yuan et al., 2026b;a; Cai et al., 2026a). The results are summarized in Table 3. For unmarked rows, we evaluate released checkpoints on LIBERO and LIBERO-PLUS using the same task configurations and random seeds as MiniWAM. Rows marked with † use externally reported results. On ROBOTWIN 2.0, we report the official clean2clean/clean2random leaderboard mean. Appendix C.4 records the source and evaluation setting of every externally reported entry.

Despite its substantially smaller scale, MiniWAM remains competitive with much larger WAMs on both benchmarks. On LIBERO, the 0.25B MiniWAM model reaches 97.2% success, comparable to the 6B-scale Fast-WAM and DreamWAM models. More importantly, PRISM retains strong performance under the distribution shifts in LIBERO-PLUS, achieving 73.6% success and outperforming Fast-WAM while approaching DreamWAM, despite using roughly an order of magnitude fewer world–action model parameters and no pretrained video-generation backbone. On ROBOTWIN 2.0,

MiniWAM reaches 46.1%, exceeding Fast-WAM and AHA-WAM. These results indicate that scaling the video model is not the only route to strong world–action learning: a compact model can remain highly competitive when its predictive objective is defined over an appropriately structured future representation.

4.4 Ablation Studies

We study how PRISM benefits from inverse-dynamics supervision and how its performance changes as the predictive bottleneck is compressed. All ablations use DINOv3 features and are evaluated on a fixed stratified 2K subset of LIBERO-Plus, covering suites, perturbation categories, and difficulty levels. Here, $n \times n$ denotes the spatial resolution of the learned bottleneck token grid, and D denotes the feature dimension of each token.

Table 4: **PRISM ablation on LIBERO-Plus-2K.** Success rate (%).

Bottleneck	Recon.	Inv. dyn.	Both
4×4, D=64	69.5	70.5	**74.1**
2×2, D=64	63.3	67.5	**67.8**
1×1, D=16	57.0	56.8	**61.7**

Table 4 reveals two consistent trends. First, neither inverse dynamics nor reconstruction alone uniformly dominates, whereas their combination performs best across all bottleneck capacities. Adding inverse-dynamics supervision to reconstruction improves success rate by roughly 4.6 percentage points across all capacities, indicating that it provides complementary control-relevant supervision beyond feature reconstruction. In Section 4.5, we further examine how this supervision shapes the learned representation geometry. Second, performance degrades as the bottleneck becomes restrictive, with PRISM success rate decreasing from 74.1% at the largest bottleneck to 61.7% at the smallest. Despite this compression, the combined PRISM objective consistently outperforms reconstruction alone.

4.5 Spatiotemporal Bottleneck Analysis

We next investigate what the spatiotemporal encoder learns under PRISM supervision. In particular, we ask whether the bottleneck reorganizes heterogeneous visual features into a compact representation structured around behaviorally relevant transitions. **Setup.** We conduct this analysis on LIBERO-90, whose 90 task IDs are disjoint from the LIBERO-40 tasks used to train the bottleneck encoder. We construct a balanced set of 4,500 transitions spanning all 90 tasks and 10 episode-progress bins, with a fixed balanced subset of 1,800 transitions used for CKA analysis. For every comparison, the native Wan-VAE and DINOv3 representations and their corresponding PRISM bottlenecks are constructed from the same physical current-to-future observation window $x_{t:t+H}$. Thus, both sides have access to identical temporal observations. The native transition features are spatially averaged at each encoder-native timestep, while the primary PRISM representation retains the full $3 \times 4 \times 4$ posterior grid L_μ. We analyze the main and wrist camera views separately.

Does PRISM align transition-specific geometry? Wan-VAE and DINOv3 encode the same visual transition with substantially different inductive biases: the former uses a generative latent space, whereas the latter provides discriminative visual features. We ask whether PRISM transforms these heterogeneous representations into a more consistent relational geometry across backbones, particularly for variation associated with the underlying transition rather than task or scene identity.

Table 5: **Cross-backbone alignment on unseen LIBERO-90 tasks.** Entries report main/wrist-view CKA.

Protocol	Raw	Random	PRISM
Global	.888/.571	.891/.469	.889/**.736**
Within-task	.372/.249	.682/.347	**.795/.705**
Deconfounded	.109/.071	.162/.162	**.379/.393**

We measure cross-backbone similarity using debiased linear centered kernel alignment (CKA) (Kornblith et al., 2019) on exactly paired transitions. Because global CKA may be dominated by task- and scene-level structure, we additionally consider two conditioned variants that reduce shared task and current-observation structure. *Global* CKA measures overall representational similarity, while *within-task* CKA subtracts each task's mean feature vector separately for each representation, then computes a single CKA score over all task-centered transitions. Finally, *task/current-deconfounded* CKA removes variation that is linearly predictable from task identity, episode progress, and the current observation

before measuring the remaining cross-backbone alignment. All nuisance models are fit using cross-fitting at the demonstration level to prevent information leakage from evaluated trajectories. Further implementation details are provided in Appendix E.2.

Table 5 shows that PRISM substantially strengthens cross-backbone alignment once task and current-state confounds are controlled. While global CKA provides little separation on the main view, PRISM markedly improves within-task and deconfounded alignment over both raw transitions and the architecture-matched random encoder. The effect is particularly pronounced for wrist observations, where within-task CKA increases from 0.249 to 0.705 and deconfounded CKA from 0.071 to 0.393. These results suggest that PRISM learns a shared transition geometry that is more consistent across fundamentally different visual backbones.

Does the learned geometry reflect behavior? Greater transition-specific agreement alone does not establish that the resulting geometry is meaningful for control. We therefore examine the local neighborhoods induced by each representation using a locked split of 882 episode-disjoint queries and 3,618 gallery transitions. For every query, we retrieve its nearest neighbors while excluding all samples from the same task. This leave-task-out protocol prevents task identity from trivially determining the retrieved examples and tests whether similar representations correspond to similar transitions across different manipulation tasks.

We first measure whether Wan- and DINO-based representations retrieve the same transitions using Jaccard similarity over their top-5 neighbor sets. Behavioral metrics are computed separately over the Wan- and DINO-retrieved top-5 neighbors and then symmetrically averaged across the two retrieval directions. Action RMSE is measured over normalized 16-step action chunks; EEF error is the Euclidean difference between current-to-future end-effector displacement vectors in metres; and gripper agreement is the fraction of action timesteps with matching gripper-command signs.

Table 6: **Behavioral structure on unseen LIBERO-90 tasks.** Leave-task-out retrieval results are reported as Raw $\rightarrow$ PRISM.

Metric	Main	Wrist
Jaccard@5 $\uparrow$	$0.170 \rightarrow$ **0.512**	$0.107 \rightarrow$ **0.333**
Action RMSE $\downarrow$	$0.400 \rightarrow$ **0.365**	$0.370 \rightarrow$ **0.272**
EEF error (m) $\downarrow$	$0.083 \rightarrow$ **0.072**	$0.089 \rightarrow$ **0.076**
Gripper agr. $\uparrow$	$0.758 \rightarrow$ **0.793**	$0.820 \rightarrow$ **0.957**

Table 6 shows that PRISM induces substantially stronger behavioral structure. In the cross-backbone comparison, Wan–DINO neighborhood agreement increases from 0.17 to 0.51 on the main view, while the retrieved transitions also become more similar in action and physical motion. The wrist view shows the same trend, with neighborhood agreement increasing from 0.11 to 0.33 and action RMSE decreasing from 0.37 to 0.27. In a separate DINOv3-only objective ablation (Appendix E.4, Table 10), full PRISM reduces action RMSE relative to reconstruction-only training from 0.414 to 0.356 on the main view and from 0.326 to 0.266 on the wrist view, with corresponding improvements in end-effector and gripper similarity. Thus, inverse-dynamics supervision contributes behavioral structure beyond compact feature reconstruction alone.

Taken together, Tables 5 and 6 show that PRISM reorganizes native visual features at both global and local scales. Across heterogeneous visual backbones, PRISM increases agreement in transition-specific geometry and produces neighborhoods that are more coherent in terms of actions and physical motion. The matched reconstruction-only comparison further shows that inverse-dynamics supervision contributes directly to this behavioral organization, rather than the effect arising from compact reconstruction alone.

5 Discussion

We presented MiniWAM, which replaces dense native future prediction with compact PRISM targets learned from current–future transitions through inverse dynamics and reconstruction. Across VAE and DINOv3 features, this improves policy performance while substantially reducing training cost, highlighting the potential of learning compact predictive targets for more efficient world–action modeling. **Limitations and future work.** PRISM is currently learned separately for each downstream setting. Future work could pretrain a shared PRISM encoder on larger and more diverse robot datasets to study transfer across tasks, environments, and embodiments. In addition, our two-stage formulation freezes PRISM before world–action training; future work could explore jointly or partially adapting the encoder in Stage 2 to further improve performance.

AI USE STATEMENT

We used generative AI tools to assist with refinement of experimental and analysis protocols, code implementation and debugging, interpretation of experimental results, and polishing of manuscript text. The authors directed the research and retain responsibility for the methodology, experimental validation, and all claims and content presented in this paper.

ETHICS STATEMENT

This work studies efficient robot policy learning using existing datasets and simulation benchmarks. Performance on these benchmarks does not establish safety or reliability in real-world environments, where distribution shifts and policy errors may cause physical harm. Deployment on physical robots requires task-specific safety validation, appropriate operational constraints, and human oversight. Use and redistribution of the datasets and pretrained models remain subject to their respective licenses and usage terms. Our planned code release is intended to support transparent evaluation and further research into reliable robot learning.

REPRODUCIBILITY STATEMENT

The main paper and appendix describe the method and experimental protocols. Section 3 specifies the two-stage learning procedure, while Appendices A–D provide architecture details, training hyperparameters, dataset and evaluation settings, and controlled-comparison protocols. Appendix E details the deterministic sampling, representation readouts, and analysis procedures. We will publicly release the implementation, experiment configurations, trained model checkpoints, and evaluation scripts upon acceptance to facilitate reproduction of our results.

REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025b. doi: 10.15607/RSS.2025.XXI.010.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to Act Anywhere with Task-centric Latent Actions. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.014.

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

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

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

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. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=itonej9GIV.

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

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 38574–38583, June 2026.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wPEIStHxYH.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

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

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

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.012.

RoboTwin Team. Robotwin 2.0 leaderboard. `https://robotwin-platform.github.io/leaderboard`, 2026. Accessed September 25, 2026.

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.

Yue Su, Sijin Chen, Haixin Shi, Mingyu Liu, Zhengshen Zhang, Ningyuan Huang, Weiheng Zhong, Zhengbang Zhu, Yuxiao Liu, and Xihui Liu. World guidance: World modeling in condition space for action generation. *arXiv preprint arXiv:2602.22010*, 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.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, et al. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026.

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.

Shanglin Yuan, Weiheng Zhao, Xin Shi, Haoyi Jiang, Xianda Guo, Liu Liu, Wenyu Liu, Wei Sui, and Xinggang Wang. Dreamwam: Beyond rgb future prediction for world action models. *arXiv preprint arXiv:2608.04996*, 2026a.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b.

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-wam: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026.

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. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=D5RNACOZEI`.

A MiniWAM and PRISM Architecture

Visual features and bottleneck. For each camera, the frozen Wan2.1 VAE receives RGB resized bilinearly to 224×224 and scaled to $[-1, 1]$. A causal encoding of the current frame and 16 future frames yields one current and four future $16 \times 28 \times 28$ latent blocks. The alternative frozen DINOv3 ViT-B/16 (LVD-1689M, revision `5931719e67bbdb9737e363e781fb0c67687896bc`) uses the pinned Hugging Face image processor at 224×224: RGB input is converted to $[0, 1]$, then the processor resizes, crops and normalizes according to its saved processor settings with rescaling disabled. We take final-layer-normalized patch tokens, excluding class and register tokens and without L2 normalization: $17 \times 196 \times 768$ for current plus 16 future frames. DINO computation is FP32 and features cross the policy boundary in BF16. Its current map is projected $768 \to 64$ by a $1 \times 1 \times 1$ convolution; the 16 future maps are projected likewise and temporally compressed by two $(3, 1, 1)$ convolutions of stride $(2, 1, 1)$, giving four future maps. No future frame is selected by subsampling.

The VAE transition grid $(5, 28, 28)$ passes through three 3D convolution–GroupNorm–SiLU blocks with output channels $(32, 64, 128)$, kernels $(1, 3, 3), (1, 3, 3), (3, 3, 3)$, strides $(1, 2, 2), (1, 2, 2), (2, 2, 2)$, and same padding. It produces $(T_L, H_L, W_L) = (3, 4, 4)$. DINO instead uses output channels $(64, 128, 128)$ and strides $(1, 2, 2), (1, 2, 2), (2, 1, 1)$ after its adapter and temporal compressor. This produces $(3, 4, 4)$ from $(5, 14, 14)$. A $1 \times 1 \times 1$ head sets $D_L = 64$. Thus $T_L \times H_L \times W_L \times D_L$ denotes temporal positions, spatial rows, spatial columns, and channels *per camera*; the default is $3 \times 4 \times 4 \times 64$, or 48 world tokens per camera. Camera-role encoders share weights within a role; camera grids are concatenated as tokens and assigned separate tiled rotary-position bands.

Feature-grid compression versus transformer sequence length. The approximately $65\times$ compression refers to the number of native future feature-grid locations replaced by PRISM targets, before the world transformer's additional patch embedding. At the default 16-frame future horizon and 224×224 image resolution, counts per camera are:

Quantity	Native Wan-VAE	Native DINOv3	PRISM
Target grid	$4 \times 28 \times 28$	$16 \times 14 \times 14$	$3 \times 4 \times 4$
Feature-grid locations	3,136	3,136	48
Channels per location	16	768	64
Target scalars	50,176	2,408,448	3,072
Future/latent transformer tokens	784	784	48

Thus PRISM reduces target-grid locations by $3136/48 \approx 65.3\times$ for either backbone. The native-prediction baselines additionally patch their feature maps with stride $(1, 2, 2)$ before the world transformer, whereas PRISM uses one token per latent location. Consequently, the reduction in future/latent *transformer tokens* is $784/48 \approx 16.3\times$, not $65\times$. Scalar-target compression is $16.3\times$ for Wan-VAE and $784\times$ for DINOv3 because channel counts differ. These counts exclude current-observation conditioning, action tokens, and text context; they do not imply the same reduction in total sequence length or wall-clock training cost. PRISM targets encode a current–future transition, whereas the native prediction targets contain the future features. The ratios here describe the default bottleneck and horizon, not every capacity or horizon ablation.

Posterior and reconstruction. Separate $1 \times 1 \times 1$ heads output μ and $\log \sigma^2$ for a diagonal Gaussian. Stage 1 uses reparameterized samples for inverse action prediction and, when enabled, reconstruction; Stage 2 freezes the trained encoder and takes μ as its deterministic target. The VAE feature decoder trilinearly upsamples the bottleneck to the future target grid, then applies $3 \times 3 \times 3$ convolutions $64 \to 64 \to 16$ with GroupNorm and SiLU between them. The DINO decoder has the same structure but outputs 768 channels at the native future patch-grid and frame resolution. Both reconstruct features, rather than RGB, and are discarded for policy inference.

Mixture of Transformers. The 0.25B model uses 12 MoT layers. Its world and action DiT experts each have width 768, FFN width 3072, 12 heads of dimension 64, and their own input/output projections, feed-forward blocks and timestep modulation. At each layer their queries, keys and values are concatenated for mixed attention, then split for expert-specific residual and feed-forward computation. Parameter totals count both experts, context projections, and active heads; the frozen visual backbone and text encoder are external feature extractors. The implementation reports approximately 247.5M policy parameters for this configuration. Native-feature arms replace the world-token projection/head with a video expert matched to the native target, so parameter counts should be compared using the run manifests rather than assumed identical.

Attention masks. Rows are queries and columns are keys; 1 means visible. Within each block attention is unrestricted. z_0 denotes clean current-observation tokens (absent from the Stage-1 inverse-dynamics MoT).

	Stage 1			Stage 2 joint		
Query \ key	z_0	L	a	z_0	L	a
z_0	—	—	—	1	0	0
L	—	1	0	1	1	1
a	—	1	1	1	1	1

In Stage 1, actions attend to the privileged L but L cannot attend to actions: the task is to infer actions from a known transition, not to predict future states. In Stage 2 joint training, L and actions attend bidirectionally. Missing-camera tokens and padded context keys are additionally masked. The fast action-only inference variant uses $[z_0, a]$, with z_0 attending only to itself and actions attending to both blocks.

B Training and Inference Details

Table 7 consolidates the common settings. An epoch refers to one pass over the corresponding extracted training windows; the LIBERO schedule is 16,260 and 54,200 optimizer updates for 15 and 50 epochs, respectively. RoboTwin uses 6 and 25 epochs. The epoch counts are the reported-run settings; the saved checkpoint configuration is the final provenance for each run.

Setting	Stage 1	Stage 2
Optimizer; peak LR; decay	AdamW; $5 \cdot 10^{-4}$; 0	same
LR schedule; warmup	cosine; 5% of updates	same
Global batch; precision; seed	256; BF16; 42	same
LIBERO duration	15 epochs	50 epochs
RoboTwin duration	6 epochs	25 epochs
LIBERO action / visual horizon	16 / 16	same
RoboTwin action / visual horizon	24 / 24	same
λ_a, λ_L	$1, 0$	$1, 1$
β; KL warmup	10^{-6}; 1,000 updates	0; none
λ_{rec}	1	0
Train/infer flow shift; steps	$5/5$; 1,000 train bins	same
Inference	—	20 denoising steps
Replan interval (environment steps)	—	LIBERO 10; RoboTwin 24

Table 7: Shared 0.25B training settings. Microbatch and gradient accumulation vary with GPU count while their product with GPU count remains 256.

Both branches use independent Gaussian noise and, in joint training, independent sampled timesteps. For $u \sim \mathcal{U}(0, 1)$, the shifted flow schedule uses $\sigma = 5u/(1 + 4u)$ and timestep 1000σ. The input is $\widetilde{y} = (1 - \sigma)y + \sigma\epsilon$ and the velocity target is $\epsilon - y$. The loss weight is a normalized, zero-minimum version of $\exp[-2(\sigma - 1/2)^2]$. Inference traverses the same shifted schedule from $u = 1$ to 0 with Euler steps. In Stage 1 the world tokens remain clean and only action flow loss is active. KL weight rises linearly from zero to β over 1,000 updates. Stage 2 jointly denoises action and world tokens, using the frozen Stage-1 encoder's posterior mean as the target. It initializes a new policy from the configured seed; the Stage-1 inverse-dynamics policy and feature decoder are not transferred.

The PRISM runs use $\lambda_{\text{rec}} = 1$ for both Wan-VAE and DINOv3. Reconstruction-only controls use weight 1. The Stage-2 decoder is inactive with $\lambda_{\text{rec}} = 0$; where retained in a checkpoint for compatibility it is frozen. The trained encoder alone supplies Stage-2 targets.

For LIBERO, a 16-action chunk is predicted every 10 environment steps. The RoboTwin policies reported in Table 3 use 24-step action and visual prediction horizons and execute the complete 24-action chunk before replanning. The separate efficiency benchmarks use visual horizons of 16 and 32 with a fixed 16-step action horizon to compare matched workloads. Initial action/world noise is sampled once per policy call and integrated for 20 inference steps. Current visual features and the instruction embedding are recomputed or retrieved for each replanning call. The frozen instruction cache uses the exact UMT5 prompt and tokenization path as online encoding; packed active BF16 rows are checked with `torch.equal` against the scalar encoder output, and vanilla LIBERO cache

matches are checked exactly. The cache is described in Appendix C. Training uses BF16 and the pinned `lwam` Conda environment: Python 3.10, PyTorch 2.11.0, Transformers 4.57.2, Accelerate 1.14.0, NumPy 1.26.4, and Einops 0.4.1. The LIBERO simulator dependencies include MuJoCo 3.3.2 and Robosuite 1.4.0. Full package pins are in `environment.yml`.

C Datasets and Evaluation Protocols

C.1 Standard LIBERO

We evaluate all four suites (`libero_10` (LIBERO-Long), `libero_spatial`, `libero_object`, `libero_goal`), ten tasks each and 50 episodes per task: 2,000 episodes total. Trial i takes the ith saved benchmark initial state. The base seed is 42; Python, NumPy, PyTorch, and environment trial seeds use $42 + i$. The policy call receives diffusion seed $42 + i$ at every replan in that trial, so the seed is reused across replans. The maximum active horizon is 700 steps for `libero_10` and 400 for the other suites, plus 30 initial no-op settling steps. Success is the environment's task-success predicate. RGB observations follow the selected visual encoder's preprocessing in Appendix A; the policy predicts normalized 16-step action chunks, denormalizes using the training-set action normalization statistics, and executes the first ten actions before replanning. The normalized action prediction is clipped to its training support before denormalization; the gripper channel is mapped to the simulator sign convention.

C.2 LIBERO-Plus

The official LIBERO-Plus fork is pinned to commit `4976dc30028e805ff8094b55501d532c48fec182`; assets use revision `dd2bd61b7d9a6fef1abc52d606e983b41886a149`, with archive SHA-256 `96764a4bfbdaea98d4411598caeab235458318fe0f549611b93d1a323027b3cf`. The full robustness evaluation contains 10,030 perturbed task IDs, one episode each. We additionally construct a fixed 2,000-task ablation subset using seed 42. The subset covers every nonempty suite $\times$ perturbation category $\times$ difficulty stratum (141 strata), allocates the remaining slots proportionally, and ranks candidates by stable SHA-256. Missing difficulty is retained as *unclassified*. We report the full population as LIBERO-Plus and the fixed 2,000-task subset as LIBERO-Plus-2K.

The text cache retains the raw fork language, a clean model-facing instruction, and the matched vanilla LIBERO instruction for auditing. For non-language perturbations, the clean string is the vanilla instruction, excluding filename-derived camera, state, texture, and layout suffixes. For the language category, the clean string preserves the deliberate paraphrase. Only the clean string is encoded. The cache checks exact BF16 tensor equivalence to online scalar UMT5 encoding and exact context/mask equivalence for mapped vanilla instructions; task IDs, text fields, revisions and hashes remain in the evaluation record. The official task-ID manifest and pinned assets determine the initial states and perturbations.

C.3 RoboTwin 2.0

The training split is the official XPolicyLab-format `demo_clean` data: 50 tasks, 50 demonstrations per task, episode IDs `0000000`–`0000049`. It uses head, left-wrist, and right-wrist RGB cameras; the two wrist-camera streams share encoder weights. Actions are 14-dimensional absolute joint positions, packed as six joints and a gripper for each arm. The official multitask evaluator is run separately under `demo_clean` and `demo_randomized`, with seed 0 and the benchmark's online valid-case/expert filter. The official leaderboard protocol evaluates 100 valid episodes per task under each of `demo_clean` and `demo_randomized`. We use the benchmark's online expert-feasibility filter to select valid evaluation cases, and report the mean task success for each condition together with their equal-weight average. Smaller frozen-manifest runs (e.g., 20 cases per task per condition, 2,000 total) are diagnostic subsets and are not substituted for the full leaderboard protocol. Frozen replay verifies recorded seeds, scenes and instructions; official mode performs online selection. The RoboTwin MiniWAM policies in Table 3 are trained with a 24-frame visual prediction horizon and a 24-step action horizon. Evaluation executes all 24 predicted actions before replanning, using the checkpoint's action and proprioception statistics. The horizon-16/32 efficiency measurements are separate profiling workloads.

C.4 Provenance of Externally Reported Baselines

Table 8 gives the per-cell sources for all entries marked with † in Table 3. Unmarked entries in the main table are our evaluations of released checkpoints. A dash means that the corresponding value is not used in the main comparison.

Table 8: **Sources of externally reported results in Table 3.** Values are success rates (%).

Method	LIBERO	LIBERO-Plus	RoboTwin 2.0
π_0-FAST	85.5 (Fei et al., 2026)	61.6 (Fei et al., 2026)	–
UniVLA	95.2 (Bu et al., 2025)	42.9 (Fei et al., 2026)	–
StarVLA-OFT	96.6 (Community, 2026)	–	24.8 (RoboTwin Team, 2026)
Xiaomi-Robotics-0	98.7 (Cai et al., 2026b)	–	40.5 (RoboTwin Team, 2026)
WorldVLA	79.1 (Cen et al., 2025; Fei et al., 2026)	25.0 (Fei et al., 2026)	–
AHA-WAM	–	–	33.8 (RoboTwin Team, 2026)

For UniVLA, we use the 42.9 reported in the archival CVPR 2026 LIBERO-Plus paper. WorldVLA reports 81.8 on standard LIBERO with its 512×512 model, but the LIBERO-Plus study evaluates the 256×256 checkpoint, which obtains 79.1 on standard LIBERO and 25.0 on LIBERO-Plus; we report this matched pair. The externally reported RoboTwin values correspond to the official clean2clean/clean2random leaderboard track, where policies are trained only on clean demonstrations. This differs from results reported by some methods under clean+randomized training. For example, AHA-WAM reports 92.8 after training on both clean and randomized data (Cai et al., 2026a), whereas the clean-trained leaderboard entry used in Table 3 is 33.8.

D Controlled Baselines and Ablations

Within each visual-feature family, all LIBERO arms use the same four-suite data, current observations, 16-step action/visual horizons, action expert width, training seed, global batch, 54,200 Stage-2 optimizer updates, and evaluation episodes. The world branch changes with its prediction target; its input/output projections and token count therefore change, and exact parameter counts are reported from each run. *Without co-training* sets the direct world-flow coefficient to zero but retains the sampled world tokens and joint attention, so the action loss can still update world-expert parameters. *Native future-feature co-training* predicts all four future Wan-VAE blocks or all 16 DINOv3 patch grids per camera, with no PRISM encoder. *PRISM co-training* predicts the frozen transition encoder's posterior-mean grid. All three share the same action objective; native and PRISM arms set world-loss weight to one.

For the Stage-1 objective ablation, reconstruction-only trains the encoder and feature decoder with feature MSE and KL but no inverse action loss; inverse-dynamics-only trains the encoder with action flow and KL but no decoder loss; inverse dynamics plus reconstruction enables both. The resulting encoders are frozen and each supplies the same Stage-2 architecture and schedule. A reconstruction-only *encoder* arm, if distinguished in a table, keeps the policy at its seeded initialization during Stage 1. Capacity variants change only the encoder bottleneck grid or channel width, with matched data and optimizer budgets. The table shorthand 4×4, 2×2, and 1×1 specifies $H_L \times W_L$; the full bottleneck is $T_L \times H_L \times W_L$ with $T_L = 3$ at the 16-frame horizon.

Efficiency measurement. The `training_core` boundary starts from resident tensors at the extracted-feature boundary and measures complete forward, backward, and optimizer updates, excluding disk I/O, data loading, collation, and host-to-device transfer. The `training_pipeline` boundary additionally includes these data-pipeline costs. Throughput is measured on one NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPU with 96 GB memory, BF16, global batch 256, and a memory-calibrated microbatch. The main-table LIBERO runs use one warmup and three measured updates, while RoboTwin runs use two warmup and six measured updates. All timings use the same pinned software environment as training.

Efficiency experiments fix the action horizon at 16 and evaluate visual horizons of 16 and 32. Stage-2 speedup is $S_{\mathrm{S2}} = t_{2,n}/t_{2,p}$ under matched horizons, camera counts, and hardware. We additionally account for the one-time PRISM Stage-1 cost. Measured Stage-1 and Stage-2 PRISM update times are nearly identical (Stage-1/Stage-2 ratios of 0.97 on LIBERO and 1.07 on ROBOTWIN), so we use a common per-update PRISM cost for this accounting. With 15/50 Stage-1/Stage-2 epochs on LIBERO and 6/25 on ROBOTWIN,

$$S_{\mathrm{total}} = \frac{S_{\mathrm{S2}}}{1 + N_1/N_2} = \begin{cases} S_{\mathrm{S2}}/1.30 & \text{LIBERO}, \\ S_{\mathrm{S2}}/1.24 & \text{ROBOTWIN}. \end{cases}$$

All speedups are computed before rounding.

DINOv3 training pipeline throughput. Table 9 complements the isolated compute results in Table 2 with throughput through the training data pipeline. Both arms use extracted visual features; PRISM additionally loads cached posterior-mean targets from its frozen encoder. Pipeline timing includes data loading, collation, host-to-device transfer, forward and backward passes, and optimizer updates. Offline feature and target extraction are excluded. The LIBERO native pipeline runs use one warmup and three measured updates; cached PRISM and RoboTwin pipeline runs use two warmup and six measured updates. LIBERO uses one main and one wrist view, while RoboTwin uses one main and two wrist views. PRISM retains 96.9–99.2% of its core throughput, with pipeline Stage-2 speedups ranging from $3.95\times$ to $9.60\times$. Thus the reduction in world-modeling compute also translates into faster updates through the data pipeline.

Table 9: **DINOv3 training throughput through the data pipeline.** Samples per second on one RTX PRO 6000 GPU at global batch 256. Core values reproduce the compute measurements in Table 2; pipeline values include data loading and transfer. Speedup compares PRISM and native Stage-2 pipeline throughput.

Benchmark	Visual horizon	Native Core	Native Pipeline	PRISM Core	PRISM Pipeline	Pipeline speedup
LIBERO	16	36.17	35.16	143.25	138.81	$3.95\times$
	32	15.55	15.26	126.09	123.08	$8.07\times$
RoboTwin 2.0	16	21.48	21.11	93.57	92.86	$4.40\times$
	32	8.61	8.41	81.47	80.74	$9.60\times$

E REPRESENTATION-ANALYSIS PROTOCOLS

E.1 COHORT AND REPRESENTATION READOUTS

We use task-indexed LIBERO-90 demonstrations, disjoint in task ID from the LIBERO-40 data used to train the analyzed transition encoders. A fixed seed-42 manifest contains 4,500 transitions: five examples in every one of the 90×10 task-by-progress cells. The nested CKA subset takes two examples per cell, for 1,800 transitions. Each row records an immutable sample ID, official task ID, source demonstration, anchor frame, and normalized progress $t/\max(1, T-1)$, where T is the episode length. The analysis verifies the sample-ID order in every source and posterior bank and uses the same paired rows at every representation stage. Main and wrist cameras are analyzed separately.

Every representation is derived from the same physical current RGB frame and 16 future RGB frames. The native Wan-VAE readout spatially averages each 16-channel latent block and concatenates one current with four causal future blocks, giving 5×16 values per transition. The native DINOv3 readout spatially averages patch features independently for the current and 16 future frames, giving 17×768 values. For each trained PRISM encoder we flatten the deterministic posterior mean L_μ in channel–time–space order, retaining the full $64 \times 3 \times 4 \times 4 = 3{,}072$ values. Thus the physical input window is matched, but native temporal resolution and feature dimension differ. Native features are spatially pooled whereas the primary PRISM readout retains its coarse spatial grid; pooled L_μ and intermediate features are sensitivity analyses.

The cross-backbone table uses the matched Wan and DINO Stage-1 inverse-dynamics-plus-reconstruction checkpoints, both with reconstruction weight $\lambda_{\text{rec}} = 1$, consistent with the policy-training configuration in Appendix B. The analysis does not mix inverse-dynamics-only posterior banks with the matched reconstruction-enabled pair.

E.2 Cross-backbone CKA

For exactly paired activation matrices $X \in \mathbb{R}^{n \times d_X}$ and $Y \in \mathbb{R}^{n \times d_Y}$, conventional linear centered kernel alignment (CKA) is

$$\mathrm{CKA}_{\text{lin}}(X, Y) = \frac{\|X_c^\top Y_c\|_F^2}{\sqrt{\|X_c^\top X_c\|_F^2 \|Y_c^\top Y_c\|_F^2}}, \tag{1}$$

where columns are centered across samples. The table reports the debiased linear estimator. With linear Gram matrices $K = XX^\top$ and $G = YY^\top$, first set their diagonals to zero and compute the U-statistic

$$\widehat{\mathrm{HSIC}}_U(K, G) = \frac{1}{n(n-3)} \left[\mathrm{tr}(KG) + \frac{(\mathbf{1}^\top K \mathbf{1})(\mathbf{1}^\top G \mathbf{1})}{(n-1)(n-2)} - \frac{2 \mathbf{1}^\top K G \mathbf{1}}{n-2} \right]. \tag{2}$$

We divide this by $\sqrt{\widehat{\mathrm{HSIC}}_U(K, K) \widehat{\mathrm{HSIC}}_U(G, G)}$. This reduces high-dimensional estimator bias: the full-grid representation has more dimensions than paired samples. Exact feature-space sufficient statistics or exact sample-Gram terms are used according to the smaller intermediate; separately normalized minibatch CKA values are never averaged. We do not standardize individual feature dimensions, because anisotropic rescaling changes the geometry. Conventional CKA is retained as a sensitivity.

We compute CKA under three distinct conditioning protocols:

- **Global.** Use all paired rows without task-specific preprocessing. Ordinary sample centering is implicit in conventional CKA; the debiased U-statistic is computed on the original rows. This retains task, scene, progress, current-observation, and transition variation.
- **Within-task.** For each official task g, separately center each backbone and stage: $X_i' = X_i - |\mathcal{I}_g|^{-1} \sum_{j \in \mathcal{I}_g} X_j$ for $i \in \mathcal{I}_g$, and analogously for Y. Compute debiased CKA over the combined residual rows. This removes task-wise mean offsets, not within-task scene variation.
- **Task/current deconfounded.** For each backbone and representation stage, predict its feature vector from official-task one-hot, normalized progress, squared progress, and that backbone's spatially pooled current-only native feature. Compute debiased CKA between the resulting held-out residual matrices. This removes only structure linearly predictable by the specified nuisance model; it does not guarantee complete removal of current-scene information.

For deconfounding, each target matrix has a separate multi-output ridge regression. Predictors and targets are standardized using training-fold statistics only. Complete demonstrations, identified by dataset and episode ID, are assigned deterministically to five approximately balanced outer folds by hashing the episode key with seed 42. Within each outer training split, three episode-grouped inner folds select one ridge coefficient from $\{10^{-4}, 10^{-2}, 1, 10^2, 10^4\}$ by mean validation MSE. A transition's nuisance prediction is therefore obtained from a model that has seen no transition from its demonstration. The same fixed paired-row cohort and fold protocol are used before and after PRISM. Backbone-specific fits are necessary because the Wan and DINO current-only features have different dimensions and scales.

The before/after comparison contrasts Wan–DINO CKA on native transition readouts with CKA on the two trained full-grid posterior means. We also evaluate seed-42 random Wan and DINO transition encoders with the same backbone-specific adapters, encoder architecture, and output grid. The random-pair CKA is computed under *all three* conditioning protocols; a trained value that does not exceed this control cannot be attributed to learned alignment. Only one trained checkpoint pair and one random initialization are analyzed, so the results are checkpoint-specific rather than estimates of training-seed variation.

Table 10: **Effect of inverse-dynamics supervision on neighborhood behavior.** Matched DINOv3 bottlenecks on unseen LIBERO-90 tasks; both retain the full posterior-mean grid.

View	Objective	Action RMSE ↓	EEF Err. (m) ↓	Grip. Agr. ↑
Main	Reconstruction only	0.414	0.090	0.747
	PRISM (I.D. + recon.)	**0.356**	**0.068**	**0.802**
Wrist	Reconstruction only	0.326	0.087	0.907
	PRISM (I.D. + recon.)	**0.266**	**0.078**	**0.975**

The primary global before/after difference has a 1,000-replicate fixed-task hierarchical bootstrap interval: all 90 tasks remain in each replicate, demonstrations are sampled with replacement within task, and selected rows from a demonstration move together. A 100-permutation null permutes DINO rows within task, applying the same permutation at the native and PRISM stages. Within-task, deconfounded, and trained-minus-random comparisons are reported as paired *point estimates*: a naive clustered percentile interval for nonlinear debiased CKA was poorly centered, while a full deconfounded interval would additionally require refitting the nested nuisance models inside each resample. These analyses neither establish causal mediation nor imply coordinate-wise equivalence.

E.3 Cross-backbone neighborhood retrieval

The 4,500-row bank is split into 882 fixed queries and 3,618 gallery transitions with no shared source demonstration between query and gallery. For each query, cosine-distance retrieval excludes every gallery transition with the same official task ID. We retrieve the top five from identical query and gallery IDs using Wan native, DINO native, Wan-PRISM full-grid, and DINO-PRISM full-grid readouts. Jaccard@5 is the per-query intersection-over-union of the Wan- and DINO-retrieved ID sets, averaged over queries. Full-grid PRISM is primary; spatially pooled PRISM is a sensitivity.

Behavioral metrics compare each query with each retrieved transition. Action RMSE uses normalized 16-step action chunks; end-effector error is the Euclidean distance between current-to-future Cartesian displacement vectors in metres; gripper agreement is the fraction of action timesteps with matching command signs. We compute each metric separately for Wan- and DINO-retrieved neighbors, then report their symmetric mean. The DINO reconstruction-only and inverse-dynamics-plus-reconstruction comparison uses identical transition IDs, query/gallery split, normalization, and retrieval rules.

E.4 DINOv3 objective ablation

Table 10 isolates the contribution of inverse-dynamics supervision using DINOv3 bottlenecks with the same latent width (64) and posterior grid ($3 \times 4 \times 4$). Both objectives reconstruct native features; full PRISM additionally uses inverse dynamics, with reconstruction weight 1. We use the same 882 queries, 3,618 gallery transitions, and leave-task-out cosine top-5 protocol described above. These are DINO-only metrics, unlike Table 6, which compares raw transitions against PRISM and averages the Wan and DINO retrieval metrics.