ON THE REPRESENTATION OF WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models (WAMs) use visual prediction to learn intermediate representations (e.g., global or dense semantics, geometry, and spatiotemporal dynamics) for robot action prediction. Learning these representations depends on which visual features are used as inputs/outputs and how these features are predicted. In this paper, we conduct comprehensive analyses on how these two choices shape intermediate representations and thus help action prediction. Our study yields two key findings: 1) V-JEPA 2.1 features (which are good at spatiotemporal dynamics) outperform Wan VAE (pixel reconstruction), DINOv3 (dense semantics), and SigLIP2 (global semantics) features for robot action prediction. 2) using frame-wise autoregression for visual feature prediction performs better than diffusion, i.e., shaping the intermediate representations better for action prediction. On RoboTwin 2.0, our representation recipe (V-JEPA 2.1 + frame-wise autoregression) achieves an average success rate of 85.9%, compared with 38.0% for the baseline recipe (Wan VAE + diffusion), without additional robot pretraining. This advantage persists after additional pre-training on 3,300 hours robot data, with success rates of 91.5% (our recipe) and 75.1% (baseline recipe). We hope these findings provide helpful guidances for developing more effective representation recipes for WAMs.

1 INTRODUCTION

Effective robot action prediction depends on understanding where objects are and how the scene can change. World action models (WAMs) use visual prediction to learn such information to help robot action prediction (Ye et al., 2026; Li et al., 2025; Yuan et al., 2026). By learning to predict future visual features, the vision expert can capture global and dense semantics, geometry, and spatiotemporal dynamics in its intermediate representations. The action expert uses these representations to predict robot actions. This connection makes representation learning central to WAMs: visual prediction is useful when it teaches the model information that the action expert can use.

A pretrained vision encoder maps current and future observations to the vision expert's input features and prediction targets, respectively. Learning these representations therefore depends on two design choices: which visual features to use and how to predict them.

In this study, we investigate how visual features and visual prediction methods shape intermediate representations for action prediction. We first consider features emphasizing pixel reconstruction, global semantics, dense semantics, and spatiotemporal dynamics, using Wan VAE (Wan Team et al., 2025), SigLIP2 (Tschannen et al., 2025), DINOv3 (Siméoni et al., 2025), and V-JEPA 2.1 (Mur-Labadia et al., 2026) as representative vision encoders, respectively. We compare the visual features to determine which categories and their corresponding vision encoders best support robot action prediction. Second, we compare frame-wise autoregression and diffusion as methods for visual feature prediction. Frame-wise autoregression predicts future visual features from preceding clean features (Zhou et al., 2025), whereas diffusion learns to recover future features from noise-corrupted inputs.

Our study yields two key findings. First, DINOv3 (dense semantics), SigLIP2 (global semantics), and V-JEPA 2.1 (Mur-Labadia et al., 2026) (spatiotemporal dynamics) features all outperform Wan VAE (pixel reconstruction) features, with V-JEPA 2.1 performing best. Second, frame-wise autoregression achieves higher robot action success rates than diffusion. Diffusion's visual- and action-

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.

loss gradients show little agreement in direction, whereas frame-wise autoregression produces more aligned gradients. These results suggest that frame-wise autoregression provides more effective visual supervision for learning intermediate representations that support action prediction.

These findings lead to our representation recipe: V-JEPA 2.1 features with frame-wise autoregression. With robot training limited to the RoboTwin 2.0 task data, our representation recipe achieves an average robot action success rate of 85.9%, compared with 38.0% for the baseline recipe (Wan VAE features with diffusion). We further test whether the advantage of this recipe holds with more robot training data. When both WAMs receive 3,300 hours of InternData (Tian et al., 2025) pretraining followed by task fine-tuning, our representation recipe still outperforms the baseline recipe by a large margin, achieving 91.5% versus 75.1% average success, which demonstrates the scalability of our representation recipe.

Our contributions are threefold:

- **Identification of effective visual features for WAMs.** We identify DINOv3, SigLIP2, and V-JEPA 2.1 features as more effective than Wan VAE features for robot action prediction, with V-JEPA 2.1 performing best. We further examine the relationship between visual feature properties and robot action success through probe evaluations and correlation analysis.
- **Discovery of the advantage of frame-wise autoregression over diffusion.** Despite the widespread use of diffusion for visual prediction in WAMs, we find that frame-wise autoregression achieves higher robot action success rates. We further investigate possible reasons for this advantage: our analyses suggest that frame-wise autoregression learns intermediate representations that better support action prediction and produces more aligned visual- and action-loss gradients.
- **A more effective WAM representation recipe with validated scalability.** We identify a more effective WAM representation recipe combining V-JEPA 2.1 features with frame-wise autoregression and validate its scalability with additional robot pretraining.

The paper is organized as follows. Section 2 describes the WAM framework and evaluation protocol. Sections 3 and 4 analyze visual features and prediction methods, respectively. Section 5 evaluates the resulting recipe with more robot pretraining data. Sections 6 and 7 discuss related work and conclude the paper.

2 Experimental Setting

2.1 WAM Framework

Our WAM uses a Mixture-of-Transformers (MoT) design (Bi et al., 2026), with a vision–action framework similar to Fast-WAM (Yuan et al., 2026). It contains separate language, vision, and action experts (Figure 1). The language expert processes the task instruction. The vision expert learns to predict future visual features. The action expert predicts actions from the intermediate representations of the vision expert. Masked attention controls which information each expert can read from the others. The language expert is initialized from Qwen3-VL and frozen; the vision and action experts are trained from random initialization. Further architecture and optimization details are provided in Appendix B.

At robot decision step t, the model receives visual observations $\mathcal{O}_t$, a task instruction g, and the robot's proprioceptive state p_t. The visual encoder E extracts features z_t, which the vision expert F_v processes into intermediate representations $\mathcal{H}_t$. The action expert uses these representations to predict an action chunk:

$$\begin{aligned} z_t &= E(\mathcal{O}_t), \\ \mathcal{H}_t &= F_v(z_t, g, p_t), \\ \mathbf{a}_t &\sim \pi_a(\cdot \mid \mathcal{H}_t, g, p_t). \end{aligned} \tag{1}$$

Here $\mathcal{H}_t$ contains the representations of the current observation at the vision expert layers that the action expert reads through masked attention. The action expert defines the conditional distribution π_a over chunks $\mathbf{a}_t = (a_t, \ldots, a_{t+H-1})$, where each a_{t+h} is a robot action and H is the prediction horizon.

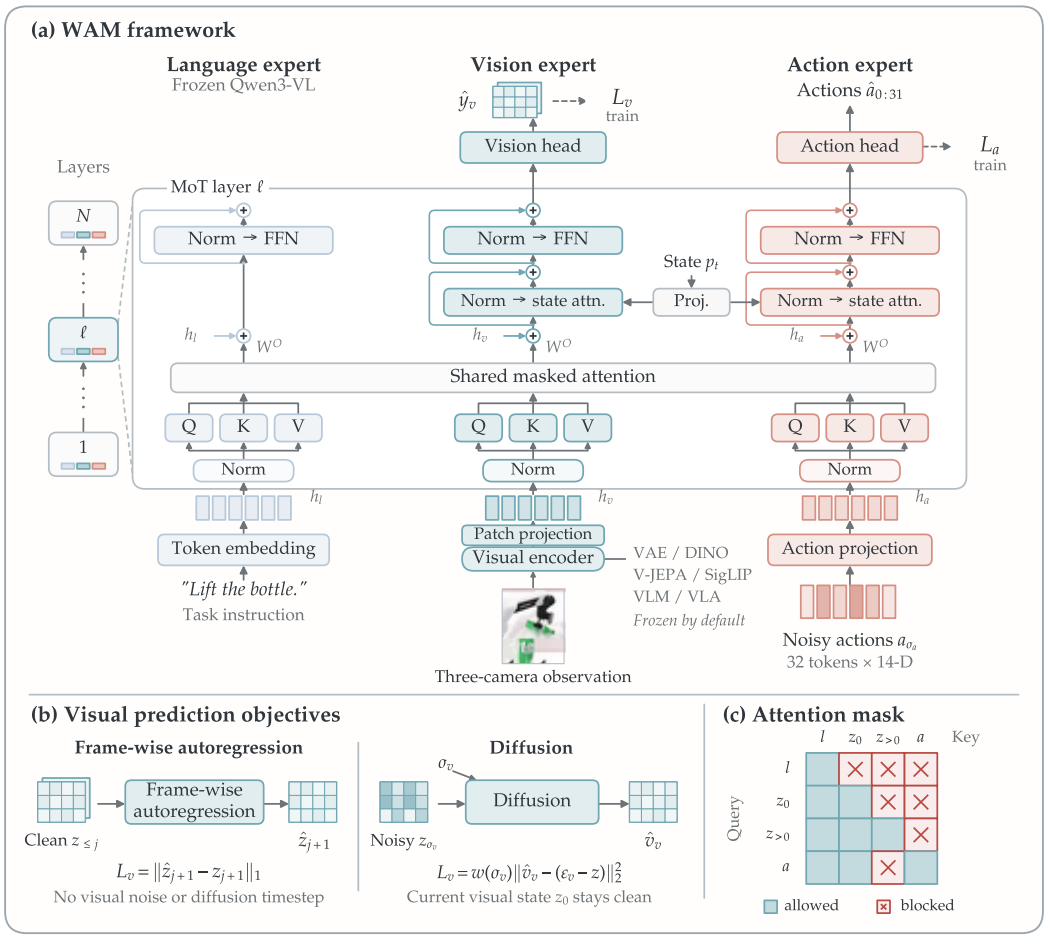


Figure 1: Our WAM framework and visual prediction methods. **(a)** Language, vision, and action experts exchange information through masked attention. The action expert uses intermediate representations from the vision expert to predict actions. **(b)** Frame-wise autoregression and diffusion train the vision expert to predict future visual features; j indexes the sequence of visual features. **(c)** Attention between instruction tokens (l), current visual tokens (z_0), future visual tokens ($z_{>0}$), and action tokens (a). Rows are queries and columns are keys.

The pretrained visual encoder E is frozen by default. It encodes current observations as inputs to the vision expert and future observations as prediction targets. Input and output projections accommodate each encoder's feature dimensions. We train the vision expert using either frame-wise autoregression or diffusion implemented through flow matching. Both methods use the same action expert. At inference, the action expert predicts actions from current-observation representations without generating future observations.

Training combines the vision prediction loss $\mathcal{L}_v$ and the action prediction loss $\mathcal{L}_a$, with weights λ_v and λ_a:

$$\mathcal{L}_{\text{WAM}} = \lambda_v \mathcal{L}_v + \lambda_a \mathcal{L}_a. \tag{2}$$

2.2 Training and Evaluation Protocol

We evaluate how visual features and visual prediction methods affect action prediction through closed-loop manipulation on RoboTwin 2.0 (Chen et al., 2025). Evaluation covers all 50 tasks, with 100 episodes per task under each of the Clean and Randomized conditions. Appendix C.1 details the training windows, camera preprocessing, and action interface, and Appendix C.3 specifies the evaluation protocol.

3 Analyses on the Input/Output Visual Representations

3.1 Visual Feature Categories

Visual features determine both what the vision expert sees and what it learns to predict. We therefore compare visual features from different vision encoders to identify which features best support robot action prediction. Our selection covers pixel reconstruction (Wan VAE), global semantics (SigLIP2), dense semantics (DINOv3), and spatiotemporal dynamics (V-JEPA 2.1), with geometry assessed across these visual features. We also consider features from further language- and robot-action training (Qwen3-VL and XR-1). During WAM training, each vision encoder remains frozen and provides both observation features and future prediction targets.

Pixel reconstruction. Wan VAE (Wan Team et al., 2025) compresses videos into latent features and learns to reconstruct the input pixels. We use its features as the baseline.

Dense semantics and spatiotemporal dynamics. DINOv3 (Siméoni et al., 2025) learns image and patch features by matching representations across augmented views of the same image. V-JEPA 2.1 (Mur-Labadia et al., 2026) learns to predict features of masked regions across video frames. This comparison contrasts image-trained DINOv3 features with video-trained V-JEPA 2.1 features.

Global semantics. SigLIP2 (Tschannen et al., 2025) learns visual features by aligning images with text. Qwen3-VL (Bai et al., 2025) initializes its vision encoder from SigLIP2 and continues training it with a language model on multimodal data. XR-1 (Xiaomi Robotics Team et al., 2026) further adapts the vision encoder of Qwen3-VL-4B-Instruct through robot-action training. We extract features from the vision encoder of each model.

Spatiotemporal dynamics from frame changes. We also test whether the vision expert can support action prediction by learning changes between frames instead of predicting future frame features. A frozen latent action model from DreamDojo (Gao et al., 2026) encodes each pair of adjacent observations as a single code describing their visual change. The vision expert predicts these codes, while a frozen V-JEPA vision encoder provides the current observation features. Appendix B.2 lists the vision encoder specifications.

3.2 Which Visual Features Are Better for Action Prediction?

We use probe experiments to evaluate global semantics, dense semantics, geometry, and robot actions from frozen visual features. We also train a WAM with each visual feature set and measure robot action success on RoboTwin 2.0. All WAM comparisons use diffusion for future visual prediction and keep the vision encoders frozen.

- **Global semantics.** Global semantics probes classify objects on ImageNet (Russakovsky et al., 2015) and scenes on SUN397 (Xiao et al., 2010).
- **Dense semantics.** Dense semantics probes predict per-pixel semantic labels on ADE20K (Zhou et al., 2017).
- **Geometry.** The Depth and Normal probes predict depth and surface orientation on NYUv2 (Silberman et al., 2012). We also match points across camera views on NAVI (Jampani et al., 2023). NAVI uses direct feature matching; the other probe experiments train prediction heads while keeping the vision encoder frozen.
- **Action probes.** We evaluate two action probes on recorded BridgeData (Ebert et al., 2022; Walke et al., 2023) demonstrations. The inverse-dynamics probe (IDM 5f) infers four actions from the five observations spanning the movement, testing whether the features describe the motion that occurred. The action-prediction probe (Action pred.) predicts the same four actions using only the first observation, testing whether the features support predicting upcoming actions. Both probes use frozen visual features and require neither WAM training nor policy rollouts.

Appendix D specifies all probe architectures, metrics, and training budgets.

Visual features / target	RoboTwin 2.0 success (%)			Global semantics		Dense semantics	Geometry			Robot actions	
	Clean	Random	Avg.	IN1k	SUN	ADE	NAVI	Depth↓	Normal↓	IDM 5f	Action pred.
Wan VAE	36.4	39.6	38.0	6.14	7.33	1.99	19.11	1.0657	56.08	0.0596	0.1950
DINOv3-S	50.5	56.0	53.3	72.91	72.01	37.11	46.65	0.4623	34.91	0.6154	0.3212
DINOv3-H+	59.1	62.3	60.7	**85.61**	80.72	**50.87**	**60.01**	0.4213	35.73	0.6880	0.3709
SigLIP2	68.9	69.0	68.9	85.57	**82.82**	42.00	42.37	0.5630	43.20	0.5997	0.3259
Qwen3-VL	64.5	63.0	63.8	83.92	81.19	15.99	19.13	0.6744	49.19	0.5368	0.3043
XR-1	70.2	69.2	69.7	75.74	78.55	18.46	19.42	0.6822	48.31	0.6573	0.3568
V-JEPA 2.1-L	**75.5**	**73.6**	**74.6**	76.85	75.45	41.68	47.30	**0.3591**	**31.47**	0.6918	0.3785
V-JEPA 2.1-G	74.9	73.6	74.2	75.52	74.80	38.34	51.12	0.4057	33.84	**0.7769**	**0.4055**
DreamDojo	71.0	69.5	70.2	N/A	N/A	N/A	N/A	N/A	N/A	0.6436	N/A
Correlation of probe performance with RoboTwin 2.0 Avg. (Spearman ρ)											
Eight visual representations				0.26	0.29	0.36	0.38	0.57	0.60	0.71	0.79

Table 1: RoboTwin 2.0 robot action success rates and probe evaluations of frozen visual features. Avg. averages Clean and Random success rates. IDM 5f and Action pred. probes report Pearson correlations between predicted and recorded actions. The bottom row reports Spearman correlations between each probe metric and RoboTwin 2.0 Avg.; Depth and Normal errors are negated so positive correlations mean that better probe scores accompany higher robot action success rates. N/A means not applicable. Bold and underlining mark the best and second-best results before rounding. Appendix D specifies probe datasets and metrics.

Findings: V-JEPA 2.1 features achieve the highest robot action success. Features emphasizing global semantics (SigLIP2), dense semantics (DINOv3), and spatiotemporal dynamics (V-JEPA 2.1) outperform pixel reconstruction features (Wan VAE).

Experimental results and analyses. Table 1 shows that DINOv3 (53.3%–60.7%), SigLIP2 (68.9%), and V-JEPA 2.1 (74.2%–74.6%) visual features all outperform Wan VAE (38.0%) on RoboTwin 2.0, with V-JEPA 2.1 performing best. V-JEPA's stronger robot action results coincide with better Depth and Normal performance, while DINOv3-H+ and SigLIP2 lead the dense and global semantics probes, respectively.

Correlation between probe performance and robot action success. We compute Spearman correlations between visual feature rankings on each probe and their rankings by average robot action success on RoboTwin 2.0. Action probes show the strongest correlations (0.71–0.79), followed by geometry probes (Depth and Normal: 0.57–0.60; NAVI: 0.38), dense semantics (0.36), and global semantics (0.26–0.29). This ordering suggests that action-related information and scene geometry encoded in visual features are more closely associated with robot action success than semantic recognition performance. The action probes exhibit the strongest correlations despite being evaluated on BridgeData, indicating that this association extends across the two datasets.

Language and action adaptation. We next ask whether visual features obtained through further language or robot-action training better support action prediction. Qwen3-VL features achieve 63.8% success on RoboTwin 2.0, below SigLIP2's 68.9%. XR-1 features reach 69.7%, a 5.9-percentage-point improvement over Qwen3-VL. In this comparison, further vision–language model training does not improve the visual features for action prediction, whereas visual features from robot-action training perform better.

Having compared the input/output visual features, we next examine the choice between frame-wise autoregression and diffusion for visual feature prediction.

4 Analyses on Visual Prediction Methods

4.1 Frame-wise Autoregression or Diffusion?

The visual prediction method shapes the intermediate representations used by the action expert. We compare frame-wise autoregression, which predicts future visual features from clean preceding features with an L1 loss, with diffusion trained through flow matching using noise-weighted velocity MSE. Both methods use the same action-prediction objective. At inference, the action expert reads current-observation representations without requiring future visual generation.

Let z_0 denote the current observation's features and $Z_{1:J} = (z_1, \ldots, z_J)$ the future features in temporal order. Conditioned on z_0, diffusion predicts a velocity from future features corrupted by Gaussian noise $\epsilon_v \sim \mathcal{N}(0, I)$ at a sampled noise level $\sigma_v \in [0, 1]$:

$$\widetilde{Z}_{1:J}(\sigma_v) = (1 - \sigma_v) Z_{1:J} + \sigma_v \epsilon_v, \qquad v_v = \epsilon_v - Z_{1:J}. \tag{3}$$

Appendices B.3 and B.4 specify the full prediction objectives, conditioning, attention masks, and noise-dependent loss weighting.

Prediction Methods	Wan VAE			V-JEPA 2.1-G		
	Clean	Random	Avg.	Clean	Random	Avg.
Diffusion	36.4	39.6	38.0	74.9	73.6	74.2
Autoregression	**74.8**	**73.0**	**73.9**	**86.4**	**85.5**	**85.9**

Table 2: RoboTwin 2.0 robot action success (%) for diffusion and frame-wise autoregression, using Wan VAE or V-JEPA 2.1-G features.

Frame-wise autoregression improves robot action success with both vision encoders (Table 2): from 74.2% to 85.9% with V-JEPA features, and from 38.0% to 73.9% with Wan VAE features. To understand this success gap, Section 4.2 examines action prediction from intermediate representations, then investigates gradient update alignment as a possible explanation for the differences between the representations.

4.2 Why Does Frame-wise Autoregression Outperform Diffusion?

We use the same V-JEPA 2.1 features for both methods to trace the observed success gap from intermediate representations to the visual supervision that shapes them. WAM weights remain fixed during these analyses. Appendix E specifies the checkpoints, data, and diagnostic procedures.

Predicting future actions from intermediate representations. We train a linear predictor to estimate future robot actions from the current-observation representations at each selected vision-expert layer. The inputs are the visual features passed through attention to the action expert, together with the initial robot state. We use the same prediction setup for every layer and method, and report action prediction error. Appendix E.2 defines the prediction targets and error metric. Figure 2(a) shows lower errors under frame-wise autoregression at all five evaluated layers, indicating that its intermediate representations better support robot action prediction.

Gradient update alignment between visual and action losses. The vision expert is updated by both the visual loss $\mathcal{L}_v$ and the action loss $\mathcal{L}_a$. To investigate why diffusion learns representations that are less useful for action prediction, we compare these losses' gradient update directions with respect to the same vision-expert parameters. For each held-out example and layer ℓ, let θ_ℓ denote the trainable parameters of that layer. Their update alignment is measured by the cosine similarity of the two gradients:

$$c_\ell = \frac{\langle \nabla_{\theta_\ell} \mathcal{L}_v, \nabla_{\theta_\ell} \mathcal{L}_a \rangle}{\|\nabla_{\theta_\ell} \mathcal{L}_v\|_2 \, \|\nabla_{\theta_\ell} \mathcal{L}_a\|_2}. \tag{4}$$

A positive value means that a gradient step reducing the visual loss also tends to reduce the action loss. A value near zero indicates little alignment between the gradient update directions; a negative value indicates opposing gradient update directions.

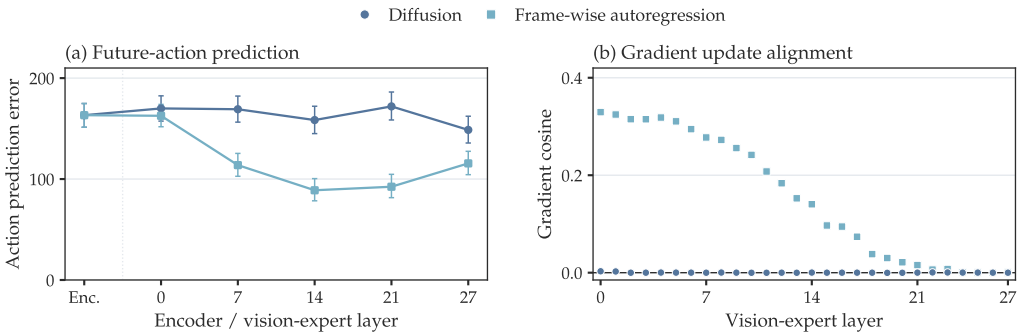


Figure 2: Comparison of frame-wise autoregression and diffusion using V-JEPA 2.1 features. **(a)** Action prediction error from vision encoder features and intermediate representations at five vision-expert layers. Error bars show 95% episode-bootstrap intervals. **(b)** Alignment between the gradient update directions induced by the visual and action losses at each vision-expert layer. Layer indices start at zero.

Frame-wise autoregression shows greater alignment between the gradient update directions, particularly in early and intermediate layers (Figure 2(b)). Averaged over the tested layers, episodes, and noise levels, the cosine similarity is 0.15445 for frame-wise autoregression and 0.00023 for diffusion. Diffusion's weaker alignment suggests that reducing its visual loss provides less direct support for reducing the action loss. This offers a possible explanation for its less useful intermediate representations and, in turn, its lower robot action success. To investigate where this weaker support may originate, we next examine a key difference between the methods: diffusion learns from noise-corrupted future features, whereas frame-wise autoregression directly predicts clean future features.

4.3 Training Sample Allocation Between Data and Noise

Noise changes what diffusion's velocity objective asks the vision expert to infer. The target is $\epsilon_v - Z_{1:J}$, and a predicted velocity $\widehat{v}_v$ defines a future-feature estimate $\widehat{Z}_{1:J} = \widetilde{Z}_{1:J}(\sigma_v) - \sigma_v \widehat{v}_v$. Its interpretation differs at the two endpoints:

1) **Near the noise endpoint** ($\sigma_v \approx 1$)**,** the input approaches ϵ_v, so velocity prediction requires recovering the future features $Z_{1:J}$.
2) **Near the data endpoint** ($\sigma_v \approx 0$)**,** the input approaches $Z_{1:J}$, so the same objective approaches noise prediction.

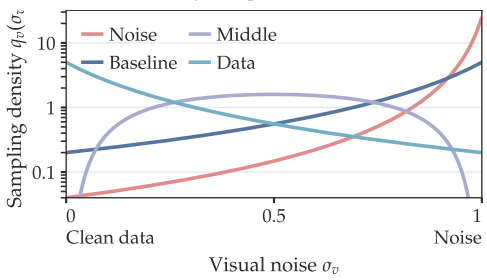
The sampling distribution therefore determines how often training emphasizes future-feature recovery or noise estimation. We examine how robot action success changes with this allocation to assess whether the role of noise helps explain diffusion's weaker support for action prediction.

In a separate set of diffusion experiments, we keep the vision encoder, velocity parameterization, and action-training schedule fixed, and vary the vision expert's training-noise preset. Each preset specifies a sampling distribution and its associated loss normalization (Appendix B.4). We compare these presets against the Baseline run in this set of experiments. Noise and Baseline concentrate training samples toward the noise endpoint, with stronger concentration for Noise. Data concentrates samples toward the data endpoint, while Middle emphasizes the middle of the interpolation. Table 3(a) plots these training sampling distributions.

Middle and Data reach 70.4% and 70.0% average robot action success, below Baseline's 72.9% (Table 3(b)). Noise reaches 72.2%. Both presets that concentrate sampling toward the noise endpoint outperform Middle and Data, while Baseline gives the highest success.

This pattern is consistent with the hypothesis that visual supervision emphasizing future-feature recovery is more useful for action prediction. Under velocity prediction, concentrating samples near the data endpoint emphasizes estimating noise from nearly clean future features; it does not turn the objective into direct future-feature prediction. Baseline also outperforms the more concentrated Noise preset, suggesting that the allocation matters beyond simply favoring the noise endpoint.

(a) Training sample allocation



(b) Robot action success (%)

Preset	Clean	Random	Avg.
Noise	73.0	71.3	72.2
Baseline	72.5	73.2	72.9
Middle	70.7	70.0	70.4
Data	70.0	70.1	70.0

Table 3: (a) Distributions of vision training samples between data and noise. (b) RoboTwin 2.0 robot action success for the corresponding vision presets, including their associated loss normalization; action training and inference schedules remain fixed.

These results characterize the tested sampling distributions together with their associated loss normalizations.

Findings: Frame-wise autoregression achieves higher robot action success than diffusion. Its more aligned visual- and action-loss gradients offer a possible explanation for why it learns intermediate representations that better support action prediction. The noise-allocation results further suggest that supervision emphasizing future-feature recovery is more useful than noise estimation near the data endpoint.

5 Scaling Up the Representation Recipe with Robot Data

The analyses in Sections 3 and 4 identify V-JEPA 2.1 features with frame-wise autoregression as the most effective recipe for robot action prediction among the configurations studied. We therefore scale up this representation recipe with additional robot pretraining data to test whether its advantage persists.

Method	**Pretraining data**	**Clean**	**Random**	**Avg.**
	w/o robot pretraining			
Wan VAE + diffusion (baseline)	0 h	36.4	39.6	38.0
V-JEPA + frame-wise autoregression (ours)	0 h	86.4	85.5	85.9
	w/ robot pretraining			
Wan VAE + diffusion (baseline)	3,300 h	76.3	74.0	75.1
V-JEPA + frame-wise autoregression (ours)	3,300 h	91.4	91.6	<u>91.5</u>
	Reported systems			
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	>10,000 h	82.7	76.8	79.8
Motus (Bi et al., 2026)	≈752k trajectories	88.7	87.0	87.9
Fast-WAM (Yuan et al., 2026)	>10,000 h† web video	91.9	91.8	91.8
LingBot-VA (Li et al., 2026)	≈16,000 h	92.9	91.6	92.2
LingBot-VA 2.0 (Zhang et al., 2026)	>16,000 h†	93.8	93.4	93.6

Table 4: RoboTwin 2.0 robot action success rate (%). Our method and baseline use the same robot-pretraining budget and RoboTwin task data. Pretraining data are robot data unless marked as web video; task data are excluded. k denotes thousands. † Educated guesses. Appendix C.4 gives the data and score sources.

We compare this recipe with the Wan VAE diffusion baseline at two robot-pretraining budgets: no additional robot pretraining, and one epoch on 3,300 hours of InternData. Both methods then use the same RoboTwin 2.0 task data. Model sizes and the RoboTwin 2.0 evaluation protocol remain fixed.

Without additional robot pretraining, V-JEPA with frame-wise autoregression achieves 85.9% average success, compared with 38.0% for the Wan VAE diffusion baseline (Table 4). With 3,300 hours of robot pretraining, our configuration reaches 91.5%, compared with 75.1% for the baseline. Its advantage is therefore maintained with more robot data, supporting the choice of V-JEPA features and frame-wise autoregression beyond training on the task data alone.

Table 4 also places these results alongside reported systems. Our model reaches 91.5% using 3,300 hours, close to Fast-WAM's reported 91.8% (Yuan et al., 2026). Appendix C.4 details the reported systems' data sources and pretraining estimates.

6 Related Work

Research on robot action prediction spans model design and visual representation learning. Vision-language-action and world action models connect visual observations with actions, while representation studies examine the information captured by pretrained vision encoders. Systematic analyses further relate world-model design choices to representation quality, providing context for how visual features and prediction methods shape intermediate representations.

Vision-language-action and world action models. OpenVLA and $\pi_{0.5}$ predict robot actions from visual observations and language instructions (Kim et al., 2024; Physical Intelligence et al., 2025). WAMs such as UVA and DreamZero jointly predict future video and actions (Li et al., 2025; Ye et al., 2026). Fast-WAM trains with visual prediction but uses the vision expert's intermediate representations to predict actions without generating future video at inference (Yuan et al., 2026).

Visual representations for embodied tasks. VC-1/CortexBench finds no pretrained representation uniformly best across embodied tasks (Majumdar et al., 2023). Probe3D evaluates geometry in frozen visual features (El Banani et al., 2024). DINO-WM plans by predicting DINOv2 features (Zhou et al., 2025), while V-JEPA 2-AC uses action-conditioned prediction of video features (Assran et al., 2025).

Systematic studies of world-model representations. Nilaksh et al. (2026) compare vision encoders under a fixed diffusion protocol, finding V-JEPA 2.1, Web-DINO, and SigLIP2 features generally more useful than pixel reconstruction features for action recovery and policy evaluation in generated environments. Yeom et al. (2026) compare video world models and pretrained vision encoders with a common inverse-dynamics probe, finding stronger action recovery from video-pretrained features despite lower pixel reconstruction fidelity. Tang et al. (2026) study WAM causal structures, latent representations, and training objectives. They find that pre-encoded temporal relations can improve in-distribution control while reducing robustness to distribution shifts, and that auxiliary-objective benefits depend on the training schedule. We focus on the intermediate representations used by a WAM's action expert, relating feature probes to robot action success and comparing frame-wise autoregression with diffusion through visual- and action-loss gradient agreement.

7 Conclusion

We studied how visual features and visual prediction methods shape intermediate representations for action prediction in WAMs. V-JEPA 2.1 features achieve higher robot action success rates than Wan VAE, DINOv3, and SigLIP2 features. Across the evaluated vision encoders, geometry probe performance correlates more strongly with robot action success than global or dense semantics performance. Frame-wise autoregression outperforms diffusion with both Wan VAE and V-JEPA 2.1 features. With V-JEPA 2.1, greater alignment between visual- and action-loss gradient update directions suggests that frame-wise autoregression provides supervision that better supports learning intermediate representations for action prediction. The noise-allocation analysis further suggests the value of supervision emphasizing future-feature recovery. The resulting recipe combines V-JEPA 2.1 features with frame-wise autoregression and retains its advantage after pretraining on 3,300 hours of robot data and task fine-tuning, achieving 91.5% average robot action success on RoboTwin 2.0 versus 75.1% for the Wan VAE + diffusion baseline. These findings provide practical guidance for choosing visual features and prediction methods when developing WAM representation recipes and scaling robot pretraining.

AI USE STATEMENT

Generative AI assisted manuscript drafting, literature organization, and the preparation of editable LaTeX conceptual diagrams and plotting code.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Hongzhe Bi et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Kevin Black, Noah Brown, Danny Driess, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

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

Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *Proceedings of Robotics: Science and Systems*, 2022. doi: 10.15607/RSS.2022.XVIII.063. URL https://www.roboticsproceedings.org/rss18/p063.html.

Mohamed El Banani, Amit Raj, Kevis-Kokitsi Maninis, Abhishek Kar, Yuanzhen Li, Michael Rubinstein, Deqing Sun, Leonidas Guibas, Justin Johnson, and Varun Jampani. Probing the 3D awareness of visual foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21795–21806, 2024. URL https://arxiv.org/abs/2404.08636.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

Varun Jampani, Kevis-Kokitsi Maninis, Andreas Engelhardt, Arjun Karpur, Karen Truong, Kyle Sargent, Stefan Popov, Andre Araujo, Ricardo Martin-Brualla, Kaushal Patel, Daniel Vlasic, Vittorio Ferrari, Ameesh Makadia, Ce Liu, Yuanzhen Li, and Howard Zhou. NAVI: Category-agnostic image collections with high-quality 3D shape and pose annotations. In *Advances in Neural Information Processing Systems*, 2023. URL https://navidataset.github.io/.

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

Lin Li et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026. URL https://arxiv.org/abs/2601.21998.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.074. URL https://www.roboticsproceedings.org/rss21/p074.html.

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? In *Advances in Neural Information Processing Systems*, volume 36, pp. 655–677, 2023. doi: 10.52202/075280-0031.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026. URL https://arxiv.org/abs/2603.14482.

Nilaksh, Saurav Jha, Artem Zholus, and Sarath Chandar. Reconstruction or semantics? what makes a latent space useful for robotic world models. *arXiv preprint arXiv:2605.06388*, 2026. URL https://arxiv.org/abs/2605.06388.

Physical Intelligence et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision*, 2012. URL https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.

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.

Chao Tang, Haoqing Wang, Zilang Cen, Weishi Mi, Wei Xia, Fangcheng Liu, Anda Cheng, Yeqing Shen, Xiaohui Cui, Xiaoyuan Zhang, Yehui Tang, and Tingguang Li. What matters in designing world action models: An empirical study. *arXiv preprint arXiv:2609.24048*, 2026. URL https://arxiv.org/abs/2609.24048.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, Yaping Li, Ping Wang, Junhao Cai, Jia Zeng, Hao Dong, and Jiangmiao Pang. InternData-A1: Pioneering high-fidelity synthetic data for pre-training generalist policy. *arXiv preprint arXiv:2511.16651*, 2025. URL https://arxiv.org/abs/2511.16651.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1723–1736. PMLR, 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

Wan Team. Wan2.2: Open and advanced large-scale video generative models. Official repository, 2025. URL https://github.com/Wan-Video/Wan2.2. Accessed September 25, 2026.

Wan Team, 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.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, 2010. doi: 10.1109/CVPR.2010.5539970. URL https://doi.org/10.1109/CVPR.2010.5539970.

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

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. URL https://arxiv.org/abs/2602.15922.

Jewon Yeom, Hanseul Kim, Jeongjae Park, Sungmok Jung, Jaejin Lee, and Taesup Kim. What makes video world model latents action-relevant: Prediction over reconstruction. *arXiv preprint arXiv:2606.07687*, 2026. URL https://arxiv.org/abs/2606.07687.

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

Qihang Zhang et al. Native video-action pretraining for generalizable robot control. *arXiv preprint arXiv:2607.08639*, 2026. URL https://arxiv.org/abs/2607.08639.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 633–641, 2017.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A Limitations and Future Work

Our scaling study is limited to 3,300 hours of robot pretraining data, so whether the recipe's advantage persists at substantially larger data scales remains open. Future work will extend the comparison to larger robot datasets and examine whether the relationship between gradient alignment and action prediction persists as training scales up.

B Model and Training Settings

B.1 Architecture and optimization

The language, vision, and action experts each have 28 transformer blocks with hidden width 2048 and 16 query heads of width 128. The language expert uses eight key/value heads and feed-forward width 6144; the vision and action experts use 16 key/value heads and feed-forward width 8192. The language expert is initialized from Qwen3-VL-2B and frozen. The vision and action experts are randomly initialized and trained jointly, while the pretrained visual encoder remains frozen.

Each expert has its own attention projections and feed-forward layers. Attention combines keys and values from the token groups permitted by the mask. The vision and action experts also receive the initial robot state through cross-attention. Table 5 lists the optimization settings.

Setting	Value
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$
Learning rate	10^{-4}, cosine decay to 10^{-6}
Weight decay / gradient clipping	10^{-2} / global norm 1.0
Precision / seed	BF16 / 42
RoboTwin training	5 epochs; global batch 1536
InternData pretraining	1 epoch; global batch 3072
Vision / action loss weights	$\lambda_v = \lambda_a = 1$

Table 5: WAM optimization settings.

B.2 Visual encoders and feature processing

Table 6 lists the visual encoders used in Table 1. Qwen3-VL features come from the 2B model's vision encoder; XR-1 uses the vision encoder of its Qwen3-VL-4B-Instruct backbone.

Encoder	Parameters	Channels	Temporal input
Wan2.2 VAE	149.63M	48	Causal history
DINOv3-S/16	∼21M	384	Single observation
DINOv3-H+/16	840.59M	1280	Single observation
SigLIP2-L/16	303.69M	1024	Single observation
Qwen3-VL vision encoder	306.24M	1024	Two-frame causal
XR-1 vision encoder	306.24M	1024	Two-frame causal
V-JEPA 2.1-L/16	∼300M	1024	Two-frame causal
V-JEPA 2.1-G/16	∼2B	1664	Two-frame causal

Table 6: Encoder parameter counts, output channels, and WAM temporal inputs. Counts include only the visual encoder. Two-frame inputs end at the observation being encoded.

The WAM receives nine observations at robot-step offsets $0, 4, \ldots, 32$. DINOv3-S encodes all nine observations independently. DINOv3-H+ retains single-observation features at offsets $0, 4, 20$, while SigLIP2 uses $0, 16, 32$. Qwen3-VL, XR-1, and both V-JEPA encoders each produce three feature states at offsets $0, 16, 32$ from the two-frame inputs $(0, 0)$, $(12, 16)$, and $(28, 32)$, respectively. Wan VAE compresses the nine observations into three causal latent states at offsets $0, 16, 32$, each using observations up to its own anchor. The first state provides the current-observation context; all subsequent states are prediction targets. Thus $J = 8$ for DINOv3-S and $J = 2$ for the other visual encoders.

V-JEPA 2.1-G produces $24 \times 20 \times 1664$ features per state; the vision expert groups 2×2 spatial positions into 120 tokens. Wan VAE states have 48 channels.

Before V-JEPA encoding, RGB values are mapped to $[0, 1]$ and normalized with mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$. The matched V-JEPA models standardize features separately for each sample, channel, and encoded state:

$$\widetilde{z}_{b,d,j,h,w} = \frac{z_{b,d,j,h,w} - \mu_{b,d,j}}{\sqrt{\nu_{b,d,j}} + 10^{-6}}, \qquad \mu_{b,d,j} = \mathrm{mean}_{h,w}\, z_{b,d,j,h,w}, \quad \nu_{b,d,j} = \mathrm{var}_{h,w}\, z_{b,d,j,h,w}. \tag{5}$$

Here b, d, j, h, w index batch, channel, time, height, and width; the mean and population variance are computed over spatial positions. Actions and proprioception use fixed dataset-specific affine normalization, inverted before execution and error reporting.

DreamDojo targets. A frozen DreamDojo LAM encodes the 32 adjacent observation pairs in a 33-state window into 32-dimensional change codes. The codes use training-set means and standard deviations for each channel. Frozen V-JEPA 2.1-G features from the current observation provide 120 context tokens after 2×2 average pooling. The vision expert predicts the change codes with flow matching, and the action expert can attend to all 32 code tokens. Both streams use independent noise and separate prediction heads. At inference, they are jointly denoised while the observation context remains fixed.

B.3 Visual prediction and attention masks

In a training window, let z_0 denote the current observation's features and $z_1, \ldots, z_J$ the subsequent visual features in temporal order. Each z_j contains features from one observation or a short sequence ending at that observation, depending on the vision encoder. The conditioning information $c = (g, p_t)$ contains the task instruction g and initial robot state p_t, as in Section 2.1.

Frame-wise autoregression. The vision expert and its prediction head, denoted by R_θ with trainable parameters θ, predict features at the next step from preceding clean features, using an L1 regression loss:

$$\begin{aligned} \widehat{z}_{j+1} &= R_\theta(z_{0:j}, c), \qquad j = 0, \ldots, J-1, \\ \mathcal{L}_v^{\mathrm{AR}} &= \frac{1}{J} \sum_{j=0}^{J-1} \mathrm{MAE}(\widehat{z}_{j+1}, z_{j+1}). \end{aligned} \tag{6}$$

Here $z_{0:j} = (z_0, \ldots, z_j)$, and MAE denotes mean absolute error. A causal attention mask prevents each prediction from reading later time steps. Training supplies the encoded ground-truth features at preceding steps.

Diffusion. Let $Z_{1:J} = (z_1, \ldots, z_J)$ collect the future features. We sample Gaussian noise $\epsilon_v \sim \mathcal{N}(0, I)$ with the same shape as $Z_{1:J}$ and a noise level $\sigma_v \in [0, 1]$ from a distribution q_v. The vision expert and its velocity-prediction head, denoted by D_θ, receive the corrupted future features from Equation 3, clean current features z_0, and context c:

$$\begin{aligned} \widehat{v}_v &= D_\theta(z_0, \widetilde{Z}_{1:J}(\sigma_v), \sigma_v, c), \\ \mathcal{L}_v^{\mathrm{diffusion}} &= \mathbb{E}_{\sigma_v, \epsilon_v}[w_v(\sigma_v)\ \mathrm{MSE}(\widehat{v}_v, \epsilon_v - Z_{1:J})]\,. \end{aligned} \tag{7}$$

The target velocity $\epsilon_v - Z_{1:J}$ is the change in features per unit increase in σ_v along the interpolation. We retain the diffusion baseline's noise-dependent loss weight w_v. Current-observation features remain clean, receive the zero-noise time condition, and are excluded from the visual loss.

The action expert uses the same flow-matching interpolation and velocity target, with independently sampled action noise and noise level. Its loss averages over valid action times and coordinates. Both visual and action losses update the vision expert.

In the frame-wise autoregression–diffusion comparison, language tokens use causal attention within the instruction. Vision tokens read language and permitted vision tokens. Action tokens read language, current-observation vision tokens, and all action tokens. Within the vision stream, frame-wise autoregression allows tokens at step j to read steps up to j. Diffusion allows current-observation tokens to read one another, while future tokens read the entire visual sequence. Spatial tokens within a permitted step can attend to one another.

B.4 Training noise and action inference

The four vision-noise presets in Section 4.3 sample positions along the data–noise interpolation as follows:

$$\begin{aligned} \sigma_v &= \frac{su}{1+(s-1)u}, \quad u \sim \mathcal{U}(0,1) \quad \text{(Noise, Baseline, Data)}, \\ \sigma_v &= \text{sigmoid}(\xi), \quad \xi \sim \mathcal{N}(0,1) \qquad \text{(Middle)}. \end{aligned} \tag{8}$$

The shift s is 25 for Noise, 5 for Baseline, and 0.2 for Data. The baseline for both vision and action training uses shifted-uniform sampling with shift $s = 5$. The vision-only sweep changes the vision preset while retaining this action setting. Network time is $\tau = T\sigma$, with $T = 1000$.

The loss weight for a noise-sampling distribution q is

$$f(\sigma) = \exp[-2(\sigma - \tfrac{1}{2})^2] - \exp(-\tfrac{1}{2}), \qquad w_q(\sigma) = \frac{f(\sigma)}{Z_{q,T} + 10^{-10}}, \qquad Z_{q,T} = \frac{1}{T}\sum_{m=0}^{T-1} f(\sigma_m). \tag{9}$$

For a shifted-uniform preset, $\sigma_m = su_m/[1 + (s-1)u_m]$ and $u_m = 1 - m/T$. For Middle, $\sigma_m = \text{sigmoid}(\Phi^{-1}((m + \frac{1}{2})/T))$, where Φ is the standard normal CDF. The weights in the vision and action losses are $w_v = w_{q_v}$ and $w_a = w_{q_a}$.

Action inference starts from a normalized Gaussian action chunk and uses ten Euler updates. At update $k = 0, \ldots, 9$,

$$u_k = 1 - \frac{k}{10}, \qquad \sigma_k = \frac{5u_k}{1+4u_k}, \qquad A^{[k+1]} = A^{[k]} + (\sigma_{k+1} - \sigma_k) v_a(A^{[k]}, T\sigma_k, c), \tag{10}$$

where v_a is the action expert's predicted velocity and c contains the instruction, current-observation representations, and initial robot state. The final noise level is $\sigma_{10} = 0$. The policy inverse-normalizes the predicted chunk and executes 24 of its 32 actions before observing again. All noise presets use this inference schedule.

C Data and Evaluation

C.1 RoboTwin data and observation windows

RoboTwin training uses 27,500 episodes across 50 tasks, with 50 Clean and 500 Randomized demonstrations per task. Episodes are split with seed 42, reserving 1% (275 episodes) for validation. All windows from an episode stay in the same partition.

A training window contains states at offsets $0, 1, \ldots, 32$, actions at offsets $0, 1, \ldots, 31$, and nine visual observations at offsets $0, 4, \ldots, 32$. At the recorded 50 Hz rate, the window spans 0.64 s. The model receives the initial proprioceptive state. Windows extending past an episode repeat its final observation and state; padded actions are excluded from the loss.

Each 480×640 camera image is resized to 240×320. Bilinear antialiased resizing then places the high-camera image in a 256×320 upper panel and the two wrist-camera images in 128×160 lower panels, forming a 384×320 canvas (height $\times$ width). RGB values are mapped to $[-1, 1]$, followed by encoder-specific input normalization. The same deterministic preprocessing is used for training and evaluation. RoboTwin actions and proprioception have 14 coordinates: six joints and one gripper coordinate for each arm.

C.2 Robot pretraining and task fine-tuning

We pretrain for one epoch on approximately 3,300 hours of InternData robot data. The same 33-state, 32-action window at its native 30 Hz rate spans $32/30 \approx 1.067$ s, with nine observations sampled at 7.5 Hz. Each arm's action has ten coordinates: three translations, a six-value rotation representation, and one gripper coordinate. For initial pose (x_0, R_0) and target pose (x_h, R_h),

$$a_h = \left[\frac{R_0^\top (x_h - x_0)}{0.25},\ r_6(R_0^\top R_h),\ 2\,\text{clip}(g_h, 0, 1) - 1 \right], \tag{11}$$

where r_6 concatenates the first two columns of a rotation matrix. Translations are expressed in the initial end-effector frame and scaled by 0.25 m. Proprioception uses absolute positions and rotations, with the preceding gripper command as its gripper state; the first command supplies this value at the episode start. Action and proprioception coordinates are clipped to $[-5, 5]$. Single-arm data use the first ten coordinates, and the remaining coordinates are masked in the loss.

For RoboTwin fine-tuning, we retain the MoT and visual weights and reinitialize the action input projection, action output head, and proprioception encoder for the 14-dimensional joint/gripper interface. Both the Wan VAE diffusion baseline and V-JEPA model trained with frame-wise autoregression use this procedure and the same RoboTwin task data.

C.3 Manipulation evaluation

We evaluate all 50 RoboTwin tasks with 100 episodes per task under each of the Clean and Randomized conditions. For each condition, we average the task success rates with equal weight across tasks. Avg. is the mean of the Clean and Randomized scores. We report success rates and percentage-point gains to one decimal place, computing both from unrounded scores; external methods retain their published scores.

C.4 Reported-system pretraining data sources

External scores in Table 4 follow LingBot-VA 2.0 (Zhang et al., 2026), except Fast-WAM (Yuan et al., 2026). Pretraining data are listed in hours or trajectories, distinguishing robot data from web video and excluding task data. The $\pi_{0.5}$ lower bound follows its extension of the π_0 training corpus (Physical Intelligence et al., 2025; Black et al., 2024). For Motus, we sum the reported robot pretraining trajectory counts; no duration is given.

Fast-WAM uses no robot pretraining but initializes its video backbone from Wan2.2-5B (Yuan et al., 2026), whose video-pretraining duration is not disclosed (Wan Team, 2025). LingBot-VA reports 92.2% success with approximately 16,000 hours of robot data (Li et al., 2026). LingBot-VA 2.0 retains the same broad data mixture and adds thousands of hours of robot demonstrations, but does not report a total duration (Zhang et al., 2026). For context, Table 4 includes educated guesses of these scales: more than 10,000 hours of web video for Fast-WAM's backbone and more than 16,000 hours of robot data for LingBot-VA 2.0. These are not author-reported totals. Reported systems use different backbones, data, and protocols.

D Encoder Probe Protocol

D.1 Frozen features and prediction heads

The probes in Table 1 evaluate global semantics, dense semantics, geometry, and robot actions using frozen encoder features. Transformer probes use final-layer, normalized patch tokens before the spatial merger or language/action projector. Static images are encoded independently; video transformers repeat each image to form their input tubelet, such as $[f, f]$ for V-JEPA. Wan VAE encodes one image using the deterministic posterior mean with its pretrained 48-channel scaling. All probes use the encoder's pretrained RGB and feature normalization.

Static inputs are 384×384, producing 24×24 features; Qwen3-VL and XR-1 instead use 448×448 inputs and 28×28 features. Channel counts are 48 for Wan VAE, 384 for DINOv3-S, 1280 for DINOv3-H+, 1664 for V-JEPA 2.1-G, and 1024 for the other encoders. Table 7 summarizes the datasets, heads, budgets, and metrics. All trainable heads use AdamW.

D.2 Global semantics, dense semantics, and geometry

Global semantics. ImageNet training samples 10% of images within each class using seed 8; evaluation samples ten official validation images per class. SUN397 uses official partition 01. Both heads use learning rate 10^{-3}, weight decay 0.004, and 0.2 epoch of warmup from 2×10^{-4} followed by cosine decay. Training uses auto-augmentation and random erasing with probability 0.25. Evaluation resizes the shorter side to $384(256/224)$ and takes a 384×384 center crop before encoder-

Probe	Data: training / evaluation	Prediction head	Budget	Metric
Objects	ImageNet-1k: stratified 10% / 10 images per class	4 attentive blocks, 16 heads; encoder feature width	5 epochs; batch 128	Top-1 accuracy (%)
Scenes	SUN397 partition 01: 19,850 / 19,850	Same as objects	10 epochs; batch 128	Top-1 accuracy (%)
Segmentation	ADE20K: 20,210 / 2,000	Channel LayerNorm + 1×1 convolution	20 epochs; batch 8	mIoU (%)
Depth / normals	NYUv2 full-derived: 24,213 / 654	3×3 convolution after upsampling	5 epochs; batch 8	Depth / angular RMSE
Correspondence	NAVI wild-image pairs; 1,000 matches per pair	Nearest-neighbor ratio matching	Direct matching	Recall at 2 cm (%)
IDM / action prediction	BridgeData: 51,178 train, 2,000 dev, 6,871 val	6 transformer blocks, width 512, 8 heads	15,995 updates; batch 16	Mean Pearson correlation

Table 7: Offline probes on frozen encoder features. Optimization and input details are given in the text.

specific resizing. Encoders run in BF16 and heads in FP32. We report the best ImageNet validation checkpoint and the fixed epoch-10 SUN397 checkpoint.

Dense semantics. ADE20K predicts 150 classes. Training applies joint image/mask scaling in $[0.5, 2]$, a 384×384 crop, and horizontal flips with probability 0.5; validation preserves aspect ratio and pads the image. The head's logits are bilinearly resized to the annotation grid. Training uses seed 8, learning rate 10^{-3}, weight decay 10^{-4}, and one warmup epoch followed by cosine decay. We report the highest validation mIoU over labeled, valid pixels.

Geometry: depth and surface normals. NYUv2 heads bilinearly upsample features fourfold before a 3×3 convolution. Depth is a weighted average of 256 bins from 0.001 to 10 m; training combines scale-invariant log-depth and multi-scale gradient losses with weights 10 and 0.5. The normal head predicts a three-dimensional vector and an uncertainty value for an angular loss. Normal targets are derived from depth with four-neighbor geometry and training-set coordinate alignment. Both heads use seed 8, learning rate 5×10^{-4}, and 0.75 epoch of warmup followed by cosine decay. We evaluate the final checkpoint. Depth RMSE is measured in metres at the predicted scale on valid depths up to 10 m; normal RMSE uses angular errors in degrees. Per-image RMSE is computed on valid pixels, then averaged across images.

Geometry: cross-view correspondence. NAVI's wild-image pairs are evaluated by direct feature matching. Features are bicubically interpolated to the quarter-resolution 3D point grid and normalized. Among points with valid positive depth, nearest/second-nearest-neighbor ratios select 1,000 correspondences per pair. Recall is the percentage whose 3D distance is below 2 cm after applying the ground-truth relative camera pose.

D.3 Robot-action probes

Each BridgeData example contains five observations and four intervening seven-dimensional actions: six translation/rotation coordinates and one gripper command. Each image is encoded independently and its features are pooled to 16×16. IDM 5f receives all five observations; Action pred. receives only the first observation and predicts the same four actions. The heads use dropout 0.1, learning rate and weight decay 10^{-4}, and SmoothL1 loss. Actions are normalized using training-partition statistics. We report the checkpoint at update 15,995. Transformer-encoder training windows are deterministically resampled across trajectory-progress bins each epoch; Wan VAE uses fixed windows. Development and validation windows are fixed for every encoder.

The score averages Pearson correlation over the six continuous action coordinates. Each coordinate's correlation uses predictions and recorded actions across all examples and four action positions. A constant prediction or target receives correlation zero. For DreamDojo's IDM probe, the four adjacent image pairs supply four global change codes, preceded by a zero origin token; probes of dense semantics and geometry use the visual encoders' feature maps.

D.4 Relation to manipulation success

For each probe, we compute Spearman correlation between visual feature rankings by probe score and by RoboTwin 2.0 average success. Depth and Normal errors are negated before ranking so higher scores consistently mean better predictions; ties receive their mean rank. Table 1 reports the comparison of eight visual representations using diffusion. The DreamDojo configuration is evaluated separately because it predicts pairwise change codes while V-JEPA supplies its observation features.

E Diagnostics of Intermediate Representations

E.1 Checkpoints and evaluation windows

We use the task-trained V-JEPA 2.1-G frame-wise autoregression and diffusion checkpoints from Table 2, both saved after 19,570 optimizer updates. They share frozen visual and language encoders and feature/action normalization. Paired comparisons use the same observations, instruction, and initial robot state.

The diagnostic data contain 275 episodes held out from WAM training. From each episode of length L, we select four 33-state windows starting at $s_k = \lfloor k(L-33)/3 \rfloor$, $k = 0, 1, 2, 3$, giving 1,100 windows. Linear probes use 165 training, 55 validation, and 55 test episodes, with all four windows assigned together.

E.2 Linear probes for future-action prediction

With WAM weights fixed, we extract current-observation attention values at vision-expert layers 0, 7, 14, 21, and 27, using only the instruction and current-vision prefix. The spatial tokens are pooled to 2×2; encoder features use the same pooling. Feature scaling, PCA to 128 components, post-PCA scaling, and target standardization are fitted separately for each representation using training episodes only.

We concatenate the 128 PCA coordinates with 14 initial-state coordinates. Ridge regression predicts 14-dimensional action displacements from the initial state at action steps 4, 16, and 32. Its unpenalized intercept and weights are fitted jointly; regularization is selected from $\{0.01, 0.1, 1, 10, 100, 1000\}$ by validation MSE in original units across all 42 outputs. Adding the initial state recovers action coordinates. The action prediction error in Figure 2(a) is test MAE over the 12 arm joints and three horizons, reported in milliradians (mrad; 10^{-3} radians).

For Figure 2(a), errors are averaged within episodes. The 95% intervals are the 2.5th and 97.5th percentiles from 5,000 bootstrap resamples of the 55 test episodes, using seed 20260906.

E.3 Visual- and action-loss gradient directions

We use one window from each of 64 episodes, assigning 16 episodes to each of the four window positions. Diffusion uses visual noise levels 0.2 and 0.8; both methods use action noise levels 0.2, 0.5, and 0.8. With seed 20260906, each window receives one Gaussian action-noise draw shared between models and reused across levels. Diffusion also receives one independent visual-noise draw reused across its levels.

At fixed model weights, we compute gradients of the native losses: visual L1 for frame-wise autoregression, and weighted flow-matching MSE for diffusion and action prediction. For each vision-expert block, gradients include all trainable parameters within that block, including attention projections and feed-forward weights. We compute the cosine in Equation 4 for each window, block, and noise condition. Figure 2(b) averages within episodes over noise conditions, then across episodes for each block; the reported overall mean also averages across all 28 blocks.