AN ACTION IS WORTH ONE PATCH: UNIFIED WORLD–ACTION MODELING WITH PATCHWAM

Anonymous authors
Paper under double-blind review

ABSTRACT

Robotic actions and their visual consequences describe the same motion: an action specifies how the robot moves, and a future frame shows where this motion leads, with the robot itself visible in the frame. We hypothesize that a predicted future frame substantially constrains the action, allowing the action to be read out of the predicted motion rather than generated by a second model. Most existing world action models nevertheless attach a separate action expert, which learns the same motion again from robot demonstrations alone. In this work, we introduce PatchWAM (Patch World-Action Model), which treats continuous actions as another type of patch through a fixed, parameter-free mapping called Action-as-Patch, so that the velocity field of a fine-tuned pretrained image generator produces the future frame and the action together. Under a linear probe, the future frame predicted by PatchWAM explains as much of the action's motion as the ground-truth future frame, and clamping the future frame to a different frame during sampling changes the generated action accordingly. Under matched data and optimization settings, PatchWAM outperforms the dual-expert control at all five RoboTwin 2.0 data scales, with gains ranging from 5.8 to 14.3 percentage points. Under other training protocols, PatchWAM reaches 96.12% on RoboTwin 2.0 with additional augmented demonstrations and 91.8% on LIBERO-Plus.

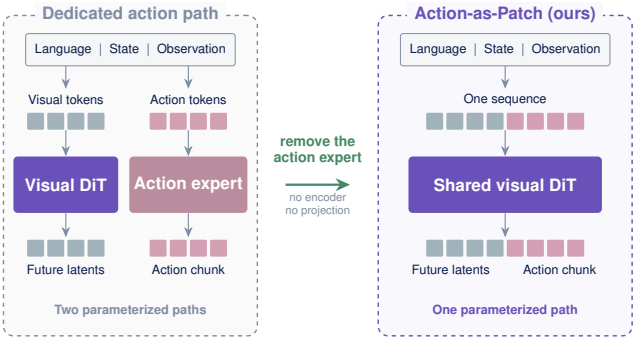


Figure 1: **Does action prediction need a dedicated DiT?** PatchWAM (right) removes the separate action expert of the dual-expert control (left) and predicts actions with fixed Action-as-Patch encoding and a shared visual DiT, so the action expert is the only difference between the two panels. Filled blocks are parameterized networks. The left panel abstracts the dual-expert control used in our comparisons rather than every dual-expert VLA; Figure 2 shows the full model.

1 INTRODUCTION

Robotic actions and their visual consequences describe the same motion: an action specifies how the robot moves, and a future frame shows where this motion leads (Ha & Schmidhuber, 2018; Kim et al., 2026; Alzayer et al., 2026). World action models (WAMs; Ye et al., 2026c; Zhang et al., 2026e; Shen et al., 2026; Wang et al., 2026b) build on this relation by predicting future frames together with actions, often with generative backbones pretrained on large image or video corpora (Rombach et al., 2022; Esser et al., 2024; Black Forest Labs, 2026; Zhang et al., 2026e). Because the robot is visible in the camera views, a predicted future frame already shows the state reached after executing the action chunk and can substantially constrain the compatible actions. This suggests that the actions could be read out of the predicted motion rather than generated by a second model.

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.

Most WAMs nevertheless attach a separate action expert (MotuBrain Team et al., 2026; Cai et al., 2026a; GigaWorld Team et al., 2026; Chen et al., 2026b; Wang et al., 2026e), as do many vision–language–action policies (Black et al., 2024; Physical Intelligence et al., 2025). Such an expert is a second generative model of the same motion with its own velocity field: its parameters are learned from robot demonstrations alone, and its prediction is not constrained to agree with the predicted future frame. These considerations motivate us to investigate whether the pretrained generator can produce the actions directly with its velocity field, together with the future frame.

We introduce PatchWAM, a Patch World-Action Model built around *Action-as-Patch* (AP), a fixed codec that maps each action step to a token in the latent space of the backbone by repetition and zero-padding (Figure 1). Action tokens therefore pass through the input projection, transformer blocks, and output projection of the backbone, without a trainable action encoder, an action-specific output projection (NVIDIA et al., 2025; Pai et al., 2025; Wang et al., 2026a), or a separate action expert; throughout this paper, a *dedicated action head* denotes a trainable action-specific prediction module beyond deterministic decoding. Conditioned on language, proprioception, and the current images, a shared diffusion transformer (DiT; Peebles & Xie, 2023) denoises the tokens of a future frame and of an action chunk at a shared noise level under a joint flow-matching objective (Lipman et al., 2022). One velocity field thus generates both, and the actions are read out without learned parameters because they are written in the token format of the backbone.

We first test the premise of this design (Section 4.3). Under a linear probe, the future frame predicted by PatchWAM explains as much of the motion within an action chunk as the ground-truth future frame, and clamping the future frame to a ground-truth frame during sampling moves the generated actions toward the state that this frame shows, whereas clamping the actions leaves the future frame unchanged. Replacing the fixed codec with a learned linear interface brings no measurable gain, so the readout needs no capacity of its own.

We then compare PatchWAM with a dual-expert control under matched data and optimization settings on RoboTwin 2.0 (Chen et al., 2025a), with 0.1% to 20% of the training windows (Section 4.4). PatchWAM achieves a higher average success rate than the control at all five data scales, and its gain is largest when the data are scarcest, which is consistent with an expert whose parameters are learned from robot demonstrations alone. With 5% of the windows, PatchWAM also reaches the final success rate of the control with fewer training steps, although both models fit the demonstrated actions about equally well. Under other training protocols, which we interpret separately from this matched control (Section 4.2), PatchWAM reaches 96.12% average success on RoboTwin 2.0 with clean, randomized, and additional augmented demonstrations, 91.8% on LIBERO-Plus (Fei et al., 2026b) with augmented training, and a RoboDojo Score of 29.39 (RoboDojo Team, 2026). Because the future-frame and action tokens are updated at every denoising step, the smaller number of trainable parameters does not imply faster inference (Appendix F).

Our contributions are threefold:

- **The action as a readout of the predicted future.** We argue that the actions need only to be read out of the predicted future frame, not generated by a second model, and we show that the predicted frame carries the actions and that the generated actions depend on it.
- **A parameter-free action interface.** Action-as-Patch writes actions in the token format of a pretrained image generator, so that one velocity field generates the future frame and the actions.
- **Empirical evaluation.** We evaluate the design with matched dual-expert comparisons at five data scales and on a second backbone, two mechanism controls, and five settings on RoboTwin 2.0, LIBERO (Liu et al., 2023), LIBERO-Plus, and RoboDojo.

2 Related Work

Vision–language–action models. Vision–language–action (VLA) models build robot policies on top of pretrained vision–language models. Some VLAs, such as OpenVLA, represent actions as discrete tokens (Kim et al., 2024), but most recent VLAs generate continuous actions through a learned action head (NVIDIA et al., 2025; Wang et al., 2026a) or a separate action expert (Black et al., 2024; Physical Intelligence et al., 2025). These modules typically learn physical dynamics primarily from robot data (Pai et al., 2025), which has motivated policies grounded in generative visual models.

World action models. World action models (WAMs) derive robot policies from video or image generation models and predict future observations jointly with actions (Wang et al., 2026b; Shen et al., 2026; Ye et al., 2026c; Zhu et al., 2025). Many WAMs retain a separate pathway for actions, in the form of a dedicated action stream or expert (MotuBrain Team et al., 2026; Cai et al., 2026a; GigaWorld Team et al., 2026; Chen et al., 2026b). Other work reduces the inference cost of future prediction by skipping it at test time (Yuan et al., 2026b; Ye et al., 2026a), distilling the model (Akbari et al., 2026), or predicting in a latent space (Luo et al., 2026a; Chen et al., 2026a). ImageWAM replaces the video model with an image editing model and predicts a single target frame (Zhang et al., 2026e); our implementation is based on ImageWAM. The controlled experiments of OpenWAM suggest that world–action synergy benefits from dedicated action capacity, explicit world-to-action information flow, and synchronized joint denoising (Wang et al., 2026e). Our model also adopts synchronized joint denoising, but we examine whether dedicated action capacity is necessary.

Actions in the visual domain. Several methods represent actions within the visual domain so that the generative backbone can process them directly. Action Images renders 7-DoF actions as multi-view action videos, so that the video backbone itself serves as the policy (Zhen et al., 2026). Hydra-0 represents actions as pixel motion across embodiments and video backbones, but maps the resulting features to executable actions with a trained action head (Li et al., 2026c). Masked Visual Actions partially reveals a trajectory in the video, so that one model can predict how the scene responds to an action or infer an action from a desired outcome (Alzayer et al., 2026). Cosmos Policy instead encodes actions as latent frames of a video model (Kim et al., 2026), and latent action models learn action-like representations from inter-frame changes (Chen et al., 2025b; Tharwat et al., 2025). Action-as-Patch also represents actions within the visual domain, but a parameter-free mapping encodes each action step as a single latent token rather than as pixels or an entire latent frame.

Joint modeling across modalities. Latent diffusion models and diffusion transformers denoise images as sequences of latent patch tokens (Rombach et al., 2022; Peebles & Xie, 2023), often trained with flow matching (Lipman et al., 2022), and recent models, including our backbone FLUX.2 [klein] (Black Forest Labs, 2026), process text and image tokens within a single sequence (Esser et al., 2024). Unified multimodal models process interleaved text, image, and video tokens in one transformer (Deng et al., 2025; Cui et al., 2025), and Cosmos 3 also includes actions (NVIDIA et al., 2026). Action-as-Patch exploits the same property within a pretrained image backbone, without a learned action tokenizer or a dedicated action expert. Jointly denoised modalities need not share a noise level: Diffusion Forcing assigns each token an independent noise level (Chen et al., 2024), and several WAMs employ distinct timesteps or schedules for video and action (Zhu et al., 2025; Guo et al., 2026; Akbari et al., 2026). We instead adopt a single noise level for action and future-frame tokens.

3 METHOD

PatchWAM encodes each action step as a single token in the native token format of the backbone, so that action prediction becomes token prediction within the existing visual path (Figure 2). Conditioned on a clean prefix of language features, a projected robot-state token, and the encoded current observation, a shared transformer jointly denoises the tokens of a future frame and of the action chunk, and a fixed decoder recovers the actions.

3.1 PROBLEM SETUP

At each time step t, the policy is conditioned on a language instruction ℓ, a multi-camera observation o_t, and a proprioceptive state s_t, and it predicts an action chunk $a_{t:t+H-1} = (a_t, \dots, a_{t+H-1}) \in \mathbb{R}^{H \times A}$, where A denotes the action dimensionality and $H = 16$ in all experiments. A world action model additionally predicts the future state of the environment (Wang et al., 2026b). In PatchWAM, this future state is the observation at the end of the chunk, $o^+ = o_{t+H}$, and the model learns the joint conditional distribution

$$p_\theta\big(o^+,\, a_{t:t+H-1} \;\big|\; \ell,\, o_t,\, s_t\big). \tag{1}$$

The ground-truth future frame is used only as a training target; at inference, the future frame and the action chunk are generated jointly, and only the actions are executed.

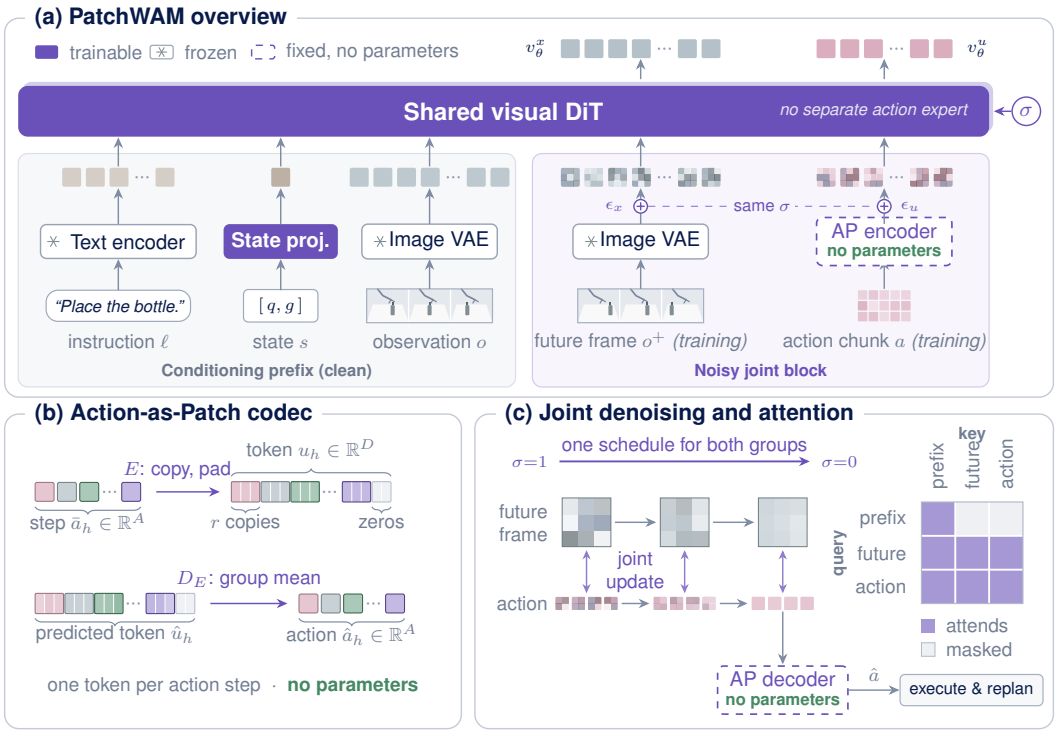


Figure 2: **Architecture of PatchWAM. (a)** Language, robot state, and the current observation form a clean conditioning prefix; the future frame and the action chunk are perturbed with independent noise at a shared noise level σ, and one shared visual DiT predicts the velocities of both groups. **(b)** Action-as-Patch repeats every action component $r = \lfloor D/A \rfloor$ times and pads with zeros, and the decoder averages the copies of each component; the codec has no parameters. **(c)** At inference, both token groups start from pure noise and are denoised together; they attend to each other and to the prefix, while the prefix does not attend to them. Token counts and camera views are schematic, and the additions on RoboDojo are not drawn.

One velocity field for both readouts. The distribution in Eq. (1) factorizes into $p(o^+ \mid \ell, o_t, s_t)$ and $p(a_{t:t+H-1} \mid o^+, \ell, o_t, s_t)$. We hypothesize that the future frame substantially constrains the second factor because the actions of a chunk move the robot from its current state s_t to the state shown in o^+; when the action is the joint state of the next step, as in two of our benchmarks, the last action $a_{t+H-1} = s_{t+H}$ is the robot state shown in o^+ itself (Appendix D). A separate action expert generates the actions with a second velocity field, whose parameters are learned from robot demonstrations alone. PatchWAM instead generates both groups of tokens with one velocity field and reads the actions out with a fixed decoder.

Tokenization. A frozen image autoencoder $\mathcal{V}$ converts each image into a sequence of latent patch tokens, $x^{\text{ref}} = \mathcal{V}(o_t)$ and $x_0 = \mathcal{V}(o^+)$, with $x^{\text{ref}}, x_0 \in \mathbb{R}^{S \times D}$, where S denotes the number of tokens and $D = 128$ the token dimension. On RoboTwin and LIBERO, the camera views are composed into a single image; on RoboDojo, each view is encoded separately. A text encoder $\mathcal{T}$ embeds the instruction as $h_\ell = \mathcal{T}(\ell)$, and a trainable linear projection maps the proprioceptive state to a single token $h_s = W_s\, s_t$. Together, these embeddings form the conditioning prefix

$$c = \left[\, h_\ell \,;\; h_s \,;\; x^{\text{ref}} \,\right]. \tag{2}$$

The fixed codec of Section 3.2 encodes the normalized action chunk into H tokens of the same dimension, $u_0 \in \mathbb{R}^{H \times D}$, so that Eq. (1) is modeled in token space as $p_\theta(x_0, u_0 \mid c)$.

Model. Given the noisy future-frame and action tokens x_σ and u_σ at a shared noise level σ, a single transformer predicts a velocity for each token group:

$$\left(v_\theta^x,\, v_\theta^u\right) = v_\theta\left(x_\sigma,\, u_\sigma,\, \sigma;\; c\right). \tag{3}$$

The trainable parameters θ comprise the transformer weights and W_s, together with the LoRA adapters used on RoboDojo; the image autoencoder and the base weights of the text encoder remain frozen, and the codec is parameter-free. The model therefore contains no action-specific network, and the gradients of the action loss reach the same transformer that denoises the future frame. The transformer is initialized from FLUX.2 [klein] 4B Base (Black Forest Labs, 2026), and the implementation is based on the ImageWAM codebase (Zhang et al., 2026e). On RoboTwin and LIBERO, $\mathcal{T}$ is Qwen3-4B (Yang et al., 2025). On RoboDojo, $\mathcal{T}$ is Qwen3-VL-4B-Instruct (Bai et al., 2025) with trainable LoRA adapters (Hu et al., 2022), and the prefix also contains past head-camera frames (Appendix B).

3.2 Action-as-Patch

Action-as-Patch maps each action step onto a token whose dimensionality matches that of a visual latent patch. Because action dimensions are expressed in heterogeneous physical units, each dimension is first normalized by a dataset-specific transform, $\bar{a} = N(a)$ (z-score on RoboTwin and RoboDojo, min–max on LIBERO), and N^{-1} is applied after decoding.

Encoder and decoder. Let $r = \lfloor D/A \rfloor$ denote the replication factor and $\alpha > 0$ a fixed scaling constant. For each step h, the encoder replicates every action component r times and zero-pads the remaining $D - rA$ entries:

$$E(\bar{a}_h) = \alpha P\, \bar{a}_h, \qquad P = \begin{bmatrix} I_A \otimes \mathbf{1}_r \\ \mathbf{0}_{(D-rA)\times A} \end{bmatrix} \in \{0,1\}^{D\times A}, \tag{4}$$

where $\otimes$ denotes the Kronecker product. Applying the encoder to each step of a chunk yields the action tokens $u_0 = \left[E(\bar{a}_1)^\top; \ldots; E(\bar{a}_H)^\top\right]$. The replication factor is $r = 9$ for the 14-dimensional RoboTwin and RoboDojo actions and $r = 18$ for the 7-dimensional LIBERO actions; in both cases, 126 entries carry action values and the remaining two are zero-padded, and $\alpha = 1$ throughout. The decoder recovers each component by averaging its group of copies:

$$[D_E(u_h)]_i = \frac{1}{\alpha r} \sum_{j=(i-1)r+1}^{ir} u_{h,j}, \qquad i = 1, \ldots, A. \tag{5}$$

Because $P^\top P = rI_A$, this readout coincides with the Moore–Penrose pseudoinverse of the encoder, $D_E(u_h) = (\alpha P)^+ u_h$. The decoder is therefore an exact left inverse of the encoder, and the codec requires no reconstruction loss. For a predicted token outside the range of E, the decoder returns the action whose encoding is nearest in the least-squares sense. The readout is therefore a fixed projection rather than a learned prediction head, and actions are never passed through the image autoencoder or quantized into a discrete vocabulary. Because an action token has the dimensionality of a latent patch, it passes through the same input and output projections as an image patch without a token-type embedding. The two groups are distinguished by their positional coordinates (Section 3.4); Appendix A discusses how replication affects the noise of each action component.

3.3 Training Objective

For each training sample, we draw a single noise level σ and two independent standard Gaussian tensors, ϵ_x and ϵ_u. We form $x_\sigma = (1-\sigma)x_0 + \sigma\epsilon_x$ and $u_\sigma = (1-\sigma)u_0 + \sigma\epsilon_u$, so the two groups share the noise level but not the noise realization. The noise level is obtained by shifting a uniform variable,

$$\sigma = \phi_s(\tau) = \frac{s\,\tau}{1 + (s-1)\,\tau}, \qquad \tau \sim \mathcal{U}(0,1), \tag{6}$$

with $s = 5$, following the schedule shift introduced for rectified-flow transformers (Esser et al., 2024); only one sixth of the sampled noise levels fall below $\sigma = \frac{1}{2}$. The transformer receives the prefix followed by the noisy tokens $[x_\sigma; u_\sigma]$ and predicts v_θ^x and v_θ^u, whose targets are $\epsilon_x - x_0$ and $\epsilon_u - u_0$. The objective is

$$\mathcal{L} = \mathbb{E}\Big[w(\sigma)\Big(\lambda_x \,\mathrm{MSE}\left(v_\theta^x, \epsilon_x - x_0\right) + \lambda_u \,\mathrm{MSE}_{\mathrm{valid}}\left(v_\theta^u, \epsilon_u - u_0\right)\Big)\Big], \tag{7}$$

where $\lambda_x = 0.5$ and $\lambda_u = 1.0$ weight the two modalities, and $w(\sigma)$ weights the noise level; it peaks at $\sigma = \frac{1}{2}$ and vanishes at $\sigma = 0$ and $\sigma = 1$ (Appendix A). Each term is averaged over

the tokens of its group before the weighted sum, so that λ_x and λ_u, rather than the token counts, set the relative weight of the two terms. The action term, $\mathrm{MSE}_{\mathrm{valid}}$, averages only over the action steps that lie within the demonstration and excludes the padded steps of chunks that extend past its end. The two zero-padded entries of each action token are retained in the regression and discarded only at decoding. On RoboDojo, the cross-entropy loss of the subtask prediction is added to $\mathcal{L}$ (Appendix B).

3.4 Token Sequence and Attention

The input sequence begins with the clean conditioning prefix c of Eq. (2), followed by the noisy joint block, in which the future-frame tokens x_σ precede the action tokens u_σ. Prefix tokens attend only to prefix tokens, whereas future-frame and action tokens attend to the prefix and to all tokens of the noisy block. Each token carries a positional coordinate with four axes. On the first axis, current-frame, future-frame, and action tokens take the values 10, 0, and 20, which only separate the groups and do not represent physical time; image tokens carry their row and column on the second and third axes, and text and action tokens carry their sequence and step indices on the fourth. Appendix C shows the token sequence and the attention mask for one RoboTwin sample.

3.5 Inference

At inference time, the model initializes the joint latent $z_0 = [x_1; u_1]$ with standard Gaussian noise, which corresponds to $\sigma_0 = 1$. It then integrates the predicted velocity with the Euler method, $z_{k+1} = z_k + (\sigma_{k+1} - \sigma_k)\, v_\theta(z_k, \sigma_k;\, c)$, along the decreasing schedule $\sigma_k = \phi_s(1 - k/K)$, $k = 0, \ldots, K-1$, which uses the same shift as in training. The default is $K = 20$ solver steps; all evaluations in this paper use $K = 10$. The final action tokens are decoded with Eq. (5) and mapped back to physical units with N^{-1}, and the robot executes the chunk, or its first part, before the policy replans from the new observation (Appendix B); the future-frame latent is not decoded into RGB.

4 Experiments

4.1 Experimental Setup

Benchmarks and training data. RoboTwin 2.0 (Chen et al., 2025a) contains 50 bimanual manipulation tasks, each evaluated in clean and randomized scenes, and we use two training protocols on this benchmark. In the full-data protocol, training uses 50 clean and 500 randomized demonstrations per task, and the strongest PatchWAM configuration adds 2,100 augmented demonstrations for seven tasks with low success rates. This protocol therefore does not measure generalization from clean to randomized scenes, and it differs from the fixed-data leaderboard protocol, which uses 50 clean demonstrations per task (RoboTwin Team, 2026). In the clean-to-random (C2R) protocol, training uses only these 50 clean demonstrations per task, as on the leaderboard, which ranks methods by their average success rate over clean and randomized scenes. The best C2R configuration applies photometric, AdaIN, and FFT-based augmentation, which is not equivalent to training on additional randomized demonstrations, and we report its checkpoint after 140k training steps. LIBERO (Liu et al., 2023) comprises four suites of ten tasks each; LIBERO-Plus (Fei et al., 2026b) perturbs these tasks, and we report it separately for training on the original LIBERO data and on augmented LIBERO-Plus data. RoboDojo (Chen et al., 2026c) spans 42 simulated tasks and five capability dimensions, and its generalization dimension is evaluated under standard and randomized conditions; we evaluate PatchWAM on all tasks and conditions.

Matched dual-expert control. The dual-expert control adds a separate action expert with 0.64B trainable parameters, initialized randomly, to the same backbone (Appendix B); at inference, it encodes the prefix once at noise level zero and denoises only the action tokens, so it does not generate a future frame. PatchWAM and the control are trained on the clean and randomized demonstrations of the full-data protocol, without the augmented demonstrations, and share all optimization settings. Both models are trained at five data scales, which retain 0.1% to 20% of the training-window starts at a fixed stride rather than a fraction of the demonstrations (Appendix E).

Table 1: **Main results on LIBERO-Plus** (SR, percent) by perturbation category. We show the strongest overall baseline, representative strong VLAs, and the strongest prior WAM; Appendix I gives the complete comparison. PatchWAM is shown in bold.

Method	Camera	Robot	Language	Light	Background	Noise	Layout	Total
QuoVLA (Wang et al., 2026c)	82.3	87.6	88.2	98.2	95.9	90.8	89.3	90.3
Anchor-Align VLA (Dalal et al., 2026)	96.3	59.1	87.2	99.0	99.6	96.9	97.4	90.8
Hermite-VLAReg (Lv et al., 2026)	89.2	85.4	87.3	97.7	96.7	93.8	89.9	91.4
ImageWAM (Zhang et al., 2026e)	79.8	58.7	95.2	96.1	91.2	93.3	83.1	85.3
PatchWAM (Ours)	**97.9**	**71.0**	**89.1**	**98.5**	**96.9**	**97.9**	**92.3**	**91.8**

Evaluation protocol. Success rates (SR) are reported in percent; on RoboTwin 2.0, the full-data and C2R evaluations run 100 episodes per task and scene condition, or 10,000 episodes in total, and each checkpoint of the matched comparisons is evaluated with 10 episodes per task and scene condition, or 1,000 episodes. RoboDojo results also include the Score, the metric by which the leaderboard ranks its entries (RoboDojo Team, 2026). All evaluations use $K = 10$ solver steps, and each result of PatchWAM comes from a single evaluation run, so no standard deviations are reported. Appendix B gives the image resolutions, batch sizes, data processing, and evaluation settings.

4.2 Results on Five Benchmark Settings

Tables 1 and 2 report the main results on LIBERO-Plus and RoboDojo; Appendix I gives complete results for all five benchmark settings. Because the baselines differ in demonstration counts, augmentation, and checkpoint selection, these results do not isolate the architectural effect, which Section 4.4 tests under matched conditions.

RoboTwin 2.0. Under the full-data protocol, PatchWAM reaches 96.04% success in clean scenes and 96.20% in randomized scenes after 110k training steps (96.12% average), 0.18 points above the strongest listed baseline, MotuBrain; this configuration includes 2,100 augmented demonstrations, whereas the run without them reaches 94.57% (Appendix H). Under the C2R protocol, PatchWAM obtains 91.56% in clean scenes and 66.72% in randomized scenes (79.14% average), the highest among the listed methods; because this configuration uses photometric, AdaIN, and FFT-based augmentation, the result cannot be attributed to the architecture alone.

LIBERO and LIBERO-Plus. On LIBERO, PatchWAM averages 98.4%, compared with 99.6% for the strongest listed baseline, QuoVLA. On LIBERO-Plus, training on the original LIBERO data gives 80.2% total success, while training on augmented LIBERO-Plus data gives 91.8%; the latter is not a zero-shot robustness result, but is 0.4 points above Hermite-VLAReg and 6.5 points above ImageWAM with its 9B backbone (Table 1).

RoboDojo. On the September 15, 2026 leaderboard snapshot, PatchWAM obtains a Score of 29.39 and an average success rate of 23.60%, compared with 24.90 and 19.34% for the strongest prior entry, DM0.5 (Table 2).[1]

4.3 Is the Action Read Out of the Predicted Future?

If the actions can be read out of the predicted future frame (Section 3.1), the frame predicted by the model should carry the actions, the generated actions should depend on it, and the readout should need no capacity of its own. We test these three consequences with the full-data model of Appendix H and two controls that are trained for the same 104,850 steps and evaluated with 100 episodes per task and condition (Table 3). In Tables 3 and 4, Δ denotes PatchWAM minus the corresponding control.

The predicted future carries the actions. Because the RoboTwin 2.0 action is the robot's joint state at the next step, repeating the current state over the chunk already explains 90% of the variance

[1] By September 22, Liber-0 Preview reported 30.74 Score and 25.52% success, and Liber-0 Lite reported 29.24 Score and 24.23% success.

Table 2: **Main results on RoboDojo.** Success rate (SR, percent) by capability dimension, followed by the leaderboard Score and average SR. Gen-Std and Gen-Rand denote generalization under standard and randomized conditions. We show representative strong entries and the strongest prior WAM from the September 15, 2026 leaderboard snapshot; Appendix I gives the complete comparison. PatchWAM is shown in bold.

Method	Gen-Std	Gen-Rand	Precision	Long	Memory	Open	Score	Avg. SR
Xiaomi-Robotics-1 (Xiaomi Robotics Team et al., 2026)	28.00	6.00	18.83	23.67	6.56	3.58	20.07	13.93
GalaxeaVLA (G0.5) (Liu et al., 2026)	21.44	4.22	20.42	32.25	7.33	1.58	20.23	14.88
DM0.5 (Dexmal Team, 2026)	17.89	4.00	16.75	19.50	47.44	2.08	24.90	19.34
OpenWAM-α (Wang et al., 2026e)	25.56	4.11	9.25	25.33	9.11	1.08	17.18	11.92
PatchWAM (Ours)	**33.33**	**16.33**	**23.00**	**31.50**	**16.67**	**22.00**	**29.39**	**23.60**

Table 3: **Mechanism controls** on RoboTwin 2.0 (SR, percent) under the full-data protocol, with 100 episodes per task and condition.

Model	Clean	Randomized	Average
PatchWAM (fixed codec)	94.68	**94.46**	**94.57**
Learned-linear codec	**95.02**	93.98	94.50
Δ	-0.34	$+0.48$	$+0.07$
PatchWAM (full attention)	**94.68**	**94.46**	**94.57**
Isolated attention	93.92	94.06	93.99
Δ	$+0.76$	$+0.40$	$+0.58$

Table 4: **Effect of a dedicated action expert** on RoboTwin 2.0 (average SR, percent) at five data scales, given as the fraction of training-window starts retained, with the number of training steps and 10 episodes per task and condition.

Windows	0.1%	1%	5%	10%	20%
Steps	8.4k	42k	42k	42k	84k
Dual-expert	16.3	75.0	78.4	82.6	87.0
PatchWAM	**30.6**	**88.3**	**88.0**	**88.4**	**92.8**
Δ	+14.3	+13.3	+9.6	+5.8	+5.8

of an action chunk, so we test the remainder, the motion within the chunk. A linear probe on autoencoder latents of the images, evaluated on held-out tasks, explains 12.2% of the variance of this motion from the current frame and 29.8% when the future frame generated by the model is added, as much as with the ground-truth future frame (30.1%); with the future frame of another window, it explains 8.5% (Appendix D).

The generated actions depend on the predicted future. When the future-frame tokens are clamped to the ground-truth frame at $t{+}8$ during sampling, the distance between the endpoint of the generated action and the robot's state at $t{+}8$ falls to 0.35 of its unclamped value and decreases in 97% of 896 windows, whereas the action of the isolated-attention control does not change. Clamping the action tokens instead leaves the generated future frame unchanged, so the dependence runs from the predicted future to the action.

The readout needs no capacity. The learned-linear control replaces the fixed codec with a trainable $A \to D \to A$ encoder–decoder pair initialized to reproduce the fixed mapping. Together, the two projections add 3,726 parameters for the 14-dimensional RoboTwin actions. The control reaches 94.50%, 0.07 points below the fixed codec, so a learned interface brings no measurable gain.

Visibility between action and future-frame tokens. The isolated-attention control prevents the noisy action and future-frame tokens from attending to one another, while each group still attends to itself and to the prefix (Appendix C). It reaches 93.99%, 0.58 points below PatchWAM, so in this data regime most of the success rate does not depend on the direct link from the predicted future to the action, and a single training run per model cannot establish even this difference. Separate noise levels for the two token groups bring no gain either (Appendix G.2).

4.4 Does Action Prediction Need a Dedicated Path?

If the actions are read out of the predicted future, a separate action expert should not be needed, and the matched comparison tests this consequence at five data scales, with the same windows, sample order, and number of training steps for both models at each scale. PatchWAM achieves a higher average success rate than the control at every scale, and its gain is largest when the data are

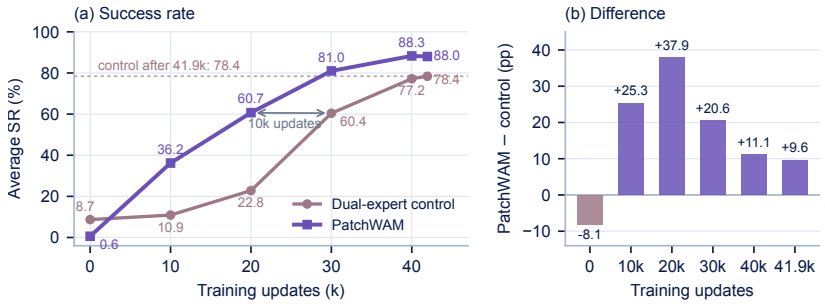


Figure 3: **Learning curves of the matched comparison at the 5% scale.** **(a)** Average RoboTwin 2.0 success rate at intermediate checkpoints; the first point is before training. **(b)** Difference in success rate (PatchWAM minus the dual-expert control). Each checkpoint is evaluated with 10 episodes per task and condition.

scarcest (Table 4). With about 42k training steps at 1%, 5%, and 10% of the windows, the success rate of the control rises from 75.0% to 82.6%, whereas that of PatchWAM stays between 88.0% and 88.4%; this pattern is consistent with an action expert whose parameters are learned from robot demonstrations alone. With Qwen-Image-2.1 (Qwen Team, 2026) as the backbone, PatchWAM likewise outperforms a dual-expert control at 0.1%, with 28.4% compared with 22.7% (Appendix E). At the 5% scale, PatchWAM also learns faster: after 30k training steps, it already exceeds the final success rate of the control, 78.4% after 41,940 steps (Figure 3; Appendix E). However, the two models also differ in token interaction and parameterization, so the gain cannot be attributed to a single factor. The gain does not come from fitting the demonstrations: at the 5% scale, the mean squared error of the generated action chunk on 1,812 training windows is 0.0034 for PatchWAM and 0.0035 for the control (Appendix D), so the two models differ in closed-loop control rather than in how well they reproduce the demonstrated actions.

5 Conclusion

PatchWAM writes continuous actions as patches so that a pretrained visual DiT jointly generates a future frame and an action chunk without an action-specific prediction module. Linear probes and clamping interventions show that the predicted frame carries information about the motion and can affect the generated action. Under matched data and optimization settings on RoboTwin 2.0, PatchWAM outperforms the dual-expert control at all five data scales, with the largest gap in the lowest-data regime; results on three benchmarks further establish its effectiveness across different training protocols.

Limitations. The matched comparisons vary action-specific capacity, token visibility, and the processing path together, and none includes independent training seeds; the full-data mechanism controls therefore do not explain every matched-comparison gap. The probes use training windows, and the isolated-attention control shows that direct future-to-action token interaction accounts for only part of the success rate. Joint denoising is also slower than the dual-expert control, and the simulation-only experiments do not establish real-time operation, physical-robot performance, or safe deployment (Appendices F and M).

Reproducibility Statement

Sections 3 and 4.1 specify the model, objectives, data, controls, and evaluation protocols. Appendices A–B give the implementation and run settings, while Appendices D, E, and H document the mechanism analyses, matched data scales, and training checkpoints. No reported result is repeated with independent training or evaluation seeds.

AI USE STATEMENT

We used generative AI to discuss the conceptual framing and experimental design, interpret results, and draft, translate, and edit the manuscript. We verified the AI-assisted work and take responsibility for the final content.

REFERENCES

Arman Akbari, Ci Zhang, Arash Akbari, Lin Zhao, Yixiao Chen, et al. Flash-WAM: Modality-aware distillation for world action models. *arXiv preprint arXiv:2606.05254*, 2026.

Hadi Alzayer, Wenlong Huang, Haonan Chen, Christopher Luey, Lvmin Zhang, et al. Masked visual actions for unified world modeling. *arXiv preprint arXiv:2607.19343*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Black Forest Labs. FLUX.2 [klein]. Official model documentation, 2026. URL https://bfl.ai/models/flux-2-klein. Accessed September 15, 2026.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, 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, 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, et al. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Hila Chefer, Patrick Esser, Dominik Lorenz, Dustin Podell, Vikash Raja, Vinh Tong, Antonio Torralba, and Robin Rombach. Self-supervised flow matching for scalable multi-modal synthesis. *arXiv preprint arXiv:2603.06507*, 2026.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, 2024.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, et al. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026a.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, et al. ABot-M0.5: Unified mobility-and-manipulation world action model. *arXiv preprint arXiv:2607.00678*, 2026b.

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*, 2025a.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Weijie Wan, Baijun Chen, Haoran Lu, Haowen Yan, Honghao Su, et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026c.

Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, et al. villa-X: Enhancing latent action modeling in vision-language-action models. *arXiv preprint arXiv:2507.23682*, 2025b.

Yizhi Chen, Zhanxiang Cao, Xinyi Peng, Yixiao Zheng, Xiaxi Si, et al. GeoAlign: Beyond semantics with state-guided spatial alignment in VLA models. *arXiv preprint arXiv:2606.03240*, 2026d.

Yufeng Cui, Honghao Chen, Haoge Deng, Xu Huang, Xinghang Li, et al. Emu3.5: Native multimodal models are world learners. *arXiv preprint arXiv:2510.26583*, 2025.

Dwip Dalal, Shivansh Patel, Chahit Jain, Jeonghwan Kim, Utkarsh Mishra, et al. Generalizable VLA finetuning via representation anchoring and language-action alignment. *arXiv preprint arXiv:2607.13429*, 2026.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Dexmal Team. DM0.5: An open-world foundation model for general-purpose embodied intelligence. Official technical blog, 2026. URL https://www.dexmal.com/blog/dm0.5/index_en.html. Accessed September 15, 2026.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the International Conference on Machine Learning*, 2024.

Liaoyuan Fan, Zetian Xu, Chen Cao, Wenyao Zhang, Mingqi Yuan, et al. AIM: Intent-aware unified world action modeling with spatial value maps. *arXiv preprint arXiv:2604.11135*, 2026.

Senyu Fei, Siyin Wang, Li Ji, Ao Li, Shiduo Zhang, et al. SRPO: Self-referential policy optimization for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a.

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*, 2026b.

GigaBrain Team, Angen Ye, Axiang Sun, Can Jin, Chenxi Cheng, et al. GigaBrain-0.7: Scaling embodied foundation models to emergent capabilities with a three-system architecture. *arXiv preprint arXiv:2608.15875*, 2026.

GigaWorld Team, Angen Ye, Angyuan Ma, Boyuan Wang, Chaojun Ni, et al. GigaWorld-Policy-0.5: A faster and stronger WAM empowered by AutoResearch. *arXiv preprint arXiv:2607.13960*, 2026.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, et al. Unified 4D world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Zeyuan He, Bowen Yang, Zhirui Fang, Keru Zhou, Lei Jiang, et al. ELAN4D: Embodiment-centric 4D supervision for vision-language-action models via plug-and-play adaptation. *arXiv preprint arXiv:2605.30484*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, et al. Galaxea open-world dataset and G0 dual-system VLA model. *arXiv preprint arXiv:2509.00576*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, et al. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, et al. Spatial Forcing: Implicit spatial representation alignment for vision-language-action model. In *International Conference on Learning Representations*, 2026a.

Haoyang Li, Guanlin Li, Youhe Feng, Chen Zhao, Zhuoran Wang, et al. Training vision-language-action models with dense embodied chain-of-thought supervision. *arXiv preprint arXiv:2606.30552*, 2026b.

Hongyu Li, Bowen Wen, Xinghao Zhu, Yixuan Wang, Yilun Du, et al. Hydra-0: Action flow for generalist world modeling and control. *arXiv preprint arXiv:2608.18077*, 2026c.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026d.

Xuewu Lin, Tianwei Lin, Yun Du, Hongyu Xie, Yiwei Jin, et al. HoloBrain-0 technical report. *arXiv preprint arXiv:2602.12062*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, et al. G0.5: One autoregressive stream for robot reasoning and action. *arXiv preprint arXiv:2608.11739*, 2026.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, et al. Being-H0.7: A latent world-action model from egocentric videos. *arXiv preprint arXiv:2605.00078*, 2026a.

Jingzhou Luo, Yifan Wen, Yongjie Bai, Xinshuai Song, Yang Liu, et al. RoVLA: Multi-consistency constraints for robust vision-language-action models. *arXiv preprint arXiv:2605.19678*, 2026b.

Yuankai Luo, Woping Chen, Tong Liang, and Zhenguo Li. CORAL: Scalable multi-task robot learning via LoRA experts. *arXiv preprint arXiv:2603.09298*, 2026c.

Yuankai Luo, Woping Chen, Tong Liang, Baiqiao Wang, and Zhenguo Li. SimVLA: A simple VLA baseline for robotic manipulation. *arXiv preprint arXiv:2602.18224*, 2026d.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, et al. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025.

Qi Lv, Jianming Xing, Zhao Yang, Mingyuan Yao, Yinan Shi, et al. Hermite curves as trajectory priors for vision-language-action models. *arXiv preprint arXiv:2608.01265*, 2026.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, et al. InternVLA-A1.5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, et al. MotuBrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026.

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

NVIDIA et al. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

NVIDIA GEAR Team, Allison Azzolini, Johan Bjorck, Valts Blukis, Fernando Castañeda, et al. GR00T N1.6: An improved open foundation model for generalist humanoid robots. NVIDIA Research project page, 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_6/. Accessed September 15, 2026.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond VLAs. *arXiv preprint arXiv:2512.15692*, 2025.

Bikang Pan, Fan Liu, Haotao Lu, Jingya Wang, and Ye Shi. SelfWAM: A self-grounded unified world action model for fast robot control. *arXiv preprint arXiv:2608.00725*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Qwen Team. Qwen-Image-2.1. Official model release, 2026. URL https://github.com/QwenLM/Qwen-Image-2.1. Accessed September 26, 2026.

RoboDojo Team. RoboDojo simulation leaderboard. Official leaderboard, 2026. URL https://robodojo-benchmark.com/leaderboard. Accessed September 15, 2026.

RoboTwin Team. RoboTwin 2.0 leaderboard. Official leaderboard and evaluation protocol, 2026. URL https://robotwin-platform.github.io/leaderboard. Accessed September 15, 2026.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, et al. World action models: A survey. *arXiv preprint arXiv:2606.20781*, 2026.

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

Bahey Tharwat, Yara Nasser, Ali Abouzeid, and Ian Reid. Latent action pretraining through world modeling. *arXiv preprint arXiv:2509.18428*, 2025.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, et al. Qwen-VLA: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026a.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, et al. World action models: The next frontier in embodied AI. *arXiv preprint arXiv:2605.12090*, 2026b.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Xuan Wang, Yinan Wu, Haoran Duan, and Jungong Han. QuoVLA: Quotient space for vision-language-action models. *arXiv preprint arXiv:2605.24890*, 2026c.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, et al. Unified vision-language-action model. In *International Conference on Learning Representations*, 2026d.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, et al. OpenWAM: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026e.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, et al. Xiaomi-Robotics-1: Scaling vision-language-action models with over 100K hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

Hengyi Xie, Chenfei Yao, Xianjin Wu, Yingying Zhu, Dingkang Liang, et al. TurboVLA: Real-time vision-language-action model at 32 Hz on an RTX 4090 with $<$1 GB VRAM. *arXiv preprint arXiv:2607.27205*, 2026a.

Zhipeng Xie, Zongyi Han, Xiangyi Wei, Shiliang Sun, Yang Li, et al. S^2-VLA: State-space guided vision-language-action models for long-horizon manipulation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2026b.

Yifu Xiong, Wenhao Yu, Jiaxuan Lin, Bojun Zou, Jiahao Li, et al. CAC-VLA: Context-gated action conditioning for vision-language-action models. *arXiv preprint arXiv:2607.04816*, 2026.

Gangwei Xu, Qihang Zhang, Jiaming Zhou, Xing Zhu, Yujun Shen, et al. Next Forcing: Causal world modeling with multi-chunk prediction. *arXiv preprint arXiv:2606.11187*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, et al. EventVLA: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026a.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, et al. 4D-WAM: Infusing spatiotemporal awareness into world action models through trajectory fields. *arXiv preprint arXiv:2608.08023*, 2026b.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, et al. ABot-M0: VLA foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026c.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, et al. GigaWorld-Policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, et al. StarVLA-α: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026b.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, et al. DreamZero: World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026c.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, et al. Qwen-RobotManip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026a.

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

He Zhang, Lingzhu Xiang, Haitao Lin, Zeyu Huang, Minghui Wang, et al. Hy-Embodied-0.5-VLA: From vision-language-action models to a real-world robot learning stack. *arXiv preprint arXiv:2606.14409*, 2026a.

Jianjun Zhang, Jian Zhu, Taiyi Su, Chong Ma, Zitai Huang, et al. Learning 4D geometric priors for inference-efficient world action models. *arXiv preprint arXiv:2607.05468*, 2026b.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, et al. Native video-action pretraining for generalizable robot control. *arXiv preprint arXiv:2607.08639*, 2026c.

Yichi Zhang, Weihao Yuan, Yizhuo Zhang, Xidong Zhang, and Jia Wan. FocusVLA: Focused visual utilization for vision-language-action models. *arXiv preprint arXiv:2603.28740*, 2026d.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, et al. ImageWAM: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026e.

Mengfei Zhao, Dihong Huang, Yikai Tang, Peihao Li, Mingxuan Yan, et al. AXIS: A growable community-driven data engine for scalable robot manipulation. *arXiv preprint arXiv:2607.21588*, 2026a.

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, et al. Faster-WAM: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026b.

Haoyu Zhen, Zixian Gao, Qiao Sun, Yilin Zhao, Yuncong Yang, et al. Action Images: End-to-end policy learning via multiview video generation. *arXiv preprint arXiv:2604.06168*, 2026.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

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

A Additional Method Details

Noise-level weight. The weight of the noise level in Eq. (7) is

$$w(\sigma) = \frac{1}{Z}\Big(\exp\big(-2\,(\sigma - \tfrac{1}{2})^2\big) - e^{-1/2}\Big). \tag{8}$$

This weight peaks at $\sigma = \frac{1}{2}$ and vanishes at $\sigma = 0$ and $\sigma = 1$, and the constant Z normalizes its expectation under Eq. (6) to one. The shift thus sets how often each noise level is sampled, and the weight sets how much each sample contributes to the loss.

Replication and noise. Replication distributes each action component across r entries of its token. Since the entries of the action noise ϵ_u are independent standard Gaussian variables, the mean of the r noisy copies of a component at noise level σ has a noise variance of σ^2/r. This redundancy therefore raises the signal-to-noise ratio of each component in the noisy input by a factor of r relative to a single copy. The same reasoning does not extend to the output: because the predicted copies of a component are not independent, averaging them does not necessarily reduce the prediction error by a fixed factor.

B Implementation Details

This appendix gives the settings of the training runs and evaluations in this paper. Table 5 lists the settings that the runs share, and Table 6 gives the number of GPUs, the batch sizes, and the number of epochs of each run. The RoboDojo run also uses past frames and a vision-language (VL) model as its text encoder, and the paragraphs *RoboDojo run* and *RoboDojo text encoder* below describe its configuration.

Inputs. On RoboTwin 2.0, the three camera views are composed into a 288×256 image, and both actions and proprioceptive states are 14-dimensional. On LIBERO, two camera views are composed horizontally into a 224×448 image, actions are 7-dimensional, and proprioceptive states are 8-dimensional. The paragraph *RoboDojo run* below describes the inputs of the RoboDojo run.

Table 5: Settings shared by the training runs. The RoboDojo run uses a different text encoder and an additional loss, as described below.

Setting	Value
Transformer	FLUX.2 [klein] 4B Base, all weights trained
State projection	linear layer, trained
Image autoencoder	FLUX.2 autoencoder, frozen
Text encoder	Qwen3-4B, frozen, 128 instruction tokens
Action chunk	$H = 16$ steps, future frame at the end of the chunk
Action-as-Patch	token width $D = 128$, scale $\alpha = 1$
Noise-level shift	$s = 5$ in training and inference
Loss weights	$\lambda_x = 0.5$, $\lambda_u = 1.0$

Training data. Each training sample is a window of 17 time steps that covers the current observation, the 16 actions of the chunk, and the observation at the end of the chunk; apart from the past frames of the RoboDojo run, only the first and the last image of the window are loaded. On RoboTwin 2.0, idle segments, in which the commanded action matches the current robot state, are removed from each demonstration before the windows are formed. Idle steps at the end of a demonstration are kept, because they often contain the release of the object. After this step, the RoboTwin 2.0 full-data set contains about 5.37 million training windows. The matched comparisons keep one window start out of every 1,000, 100, 20, 10, or 5, and Appendix E gives the number of windows and epochs at each of these data scales. The RoboDojo run keeps all steps of each demonstration. On LIBERO, the four suites are used without their no-op actions.

Table 6: Parallelism, batch size, and number of epochs of each training run. The global batch size is the product of the number of GPUs, the gradient-accumulation steps, and the batch size per GPU. The clean-to-random and RoboDojo runs stop after a fixed number of training steps instead of a number of epochs.

				Batch size		
Benchmark	Run	GPUs	Accumulation	Per GPU	Global	Epochs
RoboTwin 2.0	Full-data	32	2	4	256	5
	Full-data, augmented	32	2	4	256	5
	Matched comparisons	8	2	4	64	5–100
	Clean-to-random	16	1	16	256	—
LIBERO	Original data	16	1	8	128	10
LIBERO-Plus	Augmented data	16	2	8	256	10
RoboDojo	Past frames, VL text encoder	32	1	8	256	—

Image augmentation. All RoboTwin 2.0 runs except the clean-to-random run, as well as the RoboDojo run, augment half of the training samples online. An augmented sample is cropped to between 95% and 100% of each image side and rotated by up to 5 degrees, and it additionally receives color changes, Gaussian noise with a standard deviation of 0.01, or both. The color changes scale brightness, contrast, and saturation by factors between 0.8 and 1.2, shift the hue by up to 11 degrees, and apply a gamma between 0.9 and 1.1. All camera views and both frames of a sample share the same crop, rotation, and color parameters. In the RoboDojo run, the parameters are drawn separately for each camera view and for the past frames, and the input images of the text encoder are not augmented. LIBERO runs apply the same transforms with wider ranges, plus an exposure change, to 65% of the samples: crops to at least 92% of each side, rotations of up to 8 degrees, brightness and contrast factors between 0.7 and 1.3, saturation factors between 0.75 and 1.25, hue shifts of up to 14 degrees, gammas between 0.8 and 1.25, exposure changes of up to 0.25 stops, and noise with a standard deviation of 0.015. The clean-to-random run replaces these transforms with a stronger randomization, which is applied to 80% of the samples, with parameters drawn separately for each camera view and shared by both frames. It scales brightness, contrast, and saturation by factors between 0.6 and 1.4, shifts the hue by up to 36 degrees, applies a gamma between 0.7 and 1.4 and an exposure change of up to 0.5 stops, scales the red and blue channels by up to 15% to change the color temperature, blurs the image with probability 0.3, and adds Gaussian noise with a standard deviation of 0.03. With probability 0.5 each, it then applies an AdaIN transform, which moves the mean and standard deviation of each color channel toward random targets, and an FFT-based transform, which scales the amplitude spectrum of the image by random factors between 0.5 and 1.5 while keeping its phase. Neither of these two transforms moves edges or objects, and the run uses no crops or rotations.

RoboDojo run. The RoboDojo run keeps the three camera views as separate 256×256 images, so each view contributes 256 latent tokens per frame, and, as on RoboTwin 2.0, both actions and proprioceptive states are 14-dimensional. The future frames of all three views are predicted, and the views are distinguished by their values on the first positional axis, which are 10, 11, and 12 for the current frame and 0, 1, and 2 for the future frame. The run also conditions on 20 slots of past head-camera frames, one per second over the last 20 seconds. At inference, slot k holds the frame from k seconds before the current step, and slots that lie entirely before the start of the episode are masked. During training, the frame of each slot is drawn uniformly within 0.4 seconds of this time, the whole history is dropped with probability 0.2 and each slot with probability 0.2, and a random number of the most recent remaining slots, between 0 and 20, is kept. Each past frame is encoded by the image autoencoder, average-pooled to 4×4 tokens, and added to the prefix with the value $-k$ on the first positional axis, so the history adds 320 tokens and no parameters.

RoboDojo text encoder. The text encoder of the RoboDojo run is Qwen3-VL-4B-Instruct (Bai et al., 2025). It reads the instruction, the current head-camera image, and up to 20 past head-camera frames, one per second over the last 20 seconds, all resized to 448×448 and not augmented. During training, each past frame is shifted in time by up to one video frame; unlike the past frames of the transformer, none is dropped. The vision encoder maps each image to 196 tokens, and the tokens of

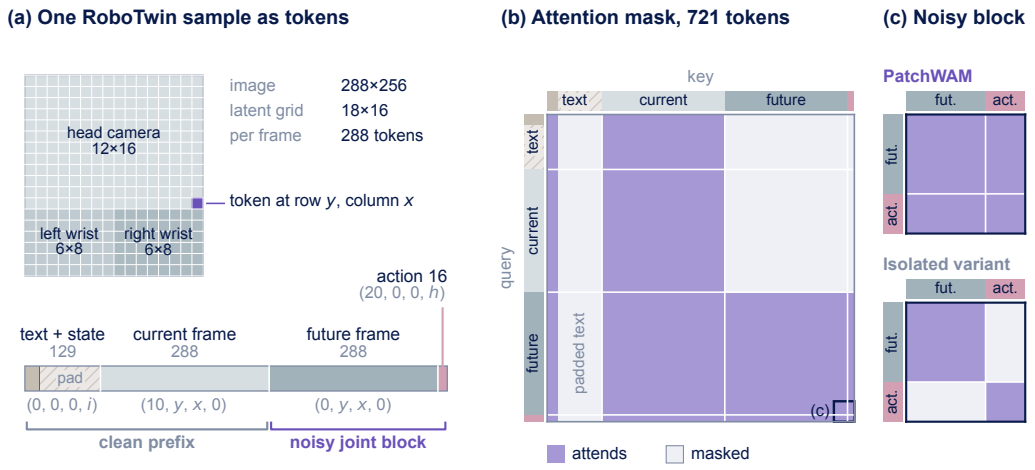


Figure 4: **Token sequence and attention mask for one RoboTwin sample.** **(a)** Each frame becomes an 18×16 grid of latent tokens; the bar shows the full sequence to scale, with the positional coordinates of each group. **(b)** The attention mask over all 721 tokens, with rows as queries and columns as keys; the strips along the edges mark the token groups of (a). **(c)** The last 48 tokens of the noisy block, boxed in (b), enlarged for PatchWAM and for the isolated-attention control of Section 4.3. The sample is taken from the first training episode, with the instruction "Grab the smooth green plastic bottle and lift it with the left arm", whose prompt occupies 25 of the 128 text tokens.

each past frame are average-pooled to 4×4 before they enter the language model. The transformer receives the concatenated hidden states of layers 9, 18, and 27 at the instruction tokens and those of layers 18, 27, and 36 at the last prompt token. The vision encoder of the VL model stays frozen. LoRA adapters (Hu et al., 2022) with rank 64, a scaling factor of 2, and dropout 0.05 are trained on the attention and feed-forward projections of its language model; the adapters add 132.1M trainable parameters. Subtask labels divide each demonstration into segments and describe each segment in one sentence, and every training demonstration carries such labels. The subtask of the current step is appended after the prompt and predicted with a cross-entropy loss of weight 0.1 and label smoothing 0.1. Because the language model attends causally, the appended subtask does not change the hidden states that the transformer receives, and the subtask is not used at inference.

Dual-expert control. The dual-expert control is ImageWAM (Zhang et al., 2026e) with the same FLUX.2 [klein] 4B backbone. Its action expert has 5 double-stream and 20 single-stream blocks of width 1024; the block counts and the attention dimensions (24 heads of size 128) match those of the backbone, so that the action tokens join the attention of the backbone in every block. The action expert embeds each action step with a linear layer, reads out the action velocity with a linear layer, and starts from random weights. It adds 0.64B trainable parameters to the model.

Evaluation. RoboTwin 2.0 evaluations use seed 43 and the unseen instructions of the benchmark, and the policy executes each predicted chunk of 16 actions in full before it predicts the next chunk from the new observation. On RoboDojo, the policy replans after executing 8 of the 16 actions. LIBERO evaluations run 20 trials per task with seed 42. On LIBERO and LIBERO-Plus, the policy predicts 16 actions and replans after executing 12 of them. All evaluations use 10 solver steps and no classifier-free guidance.

C Token Sequence and Attention Mask

Figure 4 illustrates Section 3.4 with one RoboTwin sample. The three camera views are composed into a 288×256 image, with the head camera above the left and right wrist cameras, and the image autoencoder maps this image to an 18×16 grid of latent tokens. Each frame therefore contributes 288 tokens. The instruction is padded to 128 text tokens, and the proprioceptive token is inserted directly after the valid text, so that the text block contains 129 tokens. With the two frames and the

16 action tokens, the sequence contains 721 tokens, of which the action tokens account for about 2%.

The positional coordinates follow the convention of Section 3.4. Text tokens carry their sequence index on the fourth axis, image tokens carry their row and column on the second and third axes, and action tokens carry their step index on the fourth axis. The first axis separates the current frame (10), the future frame (0), and the actions (20).

Panel (b) shows the mask used in training and inference. Prefix queries attend only to prefix keys, whereas future-frame and action queries attend to every non-padded key. Padded text tokens are masked as keys for all queries. Panel (c) contrasts this mask with the isolated-attention control, in which future-frame and action tokens no longer attend to each other but retain access to the prefix and to their own group.

D The Action as a Readout of the Predicted Future

Action and next state. On RoboTwin 2.0, we compared the action with the joint state of the next step on 31,020 transitions of 148 demonstrations, which cover all 50 tasks and both scene conditions. The two agree exactly in all 14 dimensions, including both grippers, whereas the action differs from the current joint state by the motion of one step. On RoboDojo, the training data define the action as the joint state of the next step, and the identity also holds exactly; the joint targets sent to the robot controller differ from it by the tracking error of the servos. On LIBERO, the 7-dimensional action is a normalized end-effector displacement with a gripper command, and its translation components correlate with the observed change of the end-effector position (correlation 0.90–0.99 across the four suites).

Probe. The probe uses the 896 RoboTwin 2.0 training windows of the interventions below, each with the current frame, the future frame that the full-data model generated with 10 solver steps, and the ground-truth frame at $t{+}16$. Each frame is encoded by the frozen image autoencoder, and the latents of all three camera views are reduced by PCA to 200 components per frame, fitted on the training fold only. A ridge regression, with its penalty selected by inner cross-validation, predicts the action chunk minus the current joint state repeated over the chunk, which removes the 90% of the variance that repeating the current state already explains. Table 7 reports the fraction of variance explained on held-out episodes (5 folds) and on held-out tasks (10 folds of 5 tasks), with 95% intervals of the differences from 2,000 bootstrap samples over episodes. On held-out tasks, adding the predicted future frame raises the fraction by 0.176 [0.161, 0.191], and the gain grows along the chunk from 0.122 at the first step to 0.181 at the twelfth; its difference from the ground-truth future frame is -0.003 [-0.005, 0.000]. Without removing the current state, the current frame alone reaches 0.590 and the predicted future frame raises it to 0.634 on held-out episodes, but a future frame from another episode of the same task already reaches 0.619, so the removal is needed to isolate the motion. On these training windows, the generated future frame has a root-mean-square pixel error of 12.8 on a 0–255 scale, compared with 43.0 for the current frame; on unseen episodes, the predicted frame would be less accurate.

Table 7: Fraction of the within-chunk motion explained by a linear probe on image latents (RoboTwin 2.0, 896 training windows).

Probe input	Episodes held out	Tasks held out
Current frame	0.172	0.122
Current frame + predicted future frame	0.340	0.298
Current frame + ground-truth future frame	0.341	0.301
Current frame + predicted future frame of another window	0.110	0.085
Current frame + ground-truth future frame of another episode	0.167	0.132

Interventions. During sampling, the clamped token group is replaced after every solver step by its target noised to the current level, while the other group is generated as usual; each model is evaluated on 896 windows with sampling seed 42 (Table 8). Clamping the future frame pulls the

generated action toward the state that the clamped frame shows, whereas clamping the action leaves the generated future frame unchanged. The isolated-attention control, whose action tokens cannot attend to the future-frame tokens, does not respond to any of these interventions.

Table 8: Interventions during sampling on RoboTwin 2.0 (full-data models, 896 windows). Endpoint distances are Euclidean distances in normalized joint-state space divided by the corresponding distance without clamping; square brackets denote 95% intervals. In the last two rows, parentheses contain the corresponding value without clamping.

Clamped group	Measure	PatchWAM	Isolated attention
Future frame, ground truth at $t{+}8$	Relative endpoint distance to the state at $t{+}8$	0.35 [0.33, 0.37]	1.00
Future frame, ground truth at $t{+}32$	Relative endpoint distance to the state at $t{+}32$	0.58 [0.56, 0.60]	1.00
Future frame of another episode	Action-distance ratio: other episode / original window	4.4 (49.0)	51.4 (51.4)
Action, next ground-truth chunk	Future-frame pixel MSE at $t{+}16$ (0–255 RGB)	175 (173)	159 (159)

Open-loop fit of the matched comparison. On 1,812 windows of the training data, which cover all 50 tasks and both conditions, the mean squared error between the generated and the demonstrated action chunk in normalized units is 0.0034 [0.0029, 0.0039] for PatchWAM and 0.0035 [0.0028, 0.0046] for the dual-expert control after 41,940 training steps at the 5% scale, with 95% intervals in brackets; the medians are 0.00037 and 0.00048. Repeating the current state gives 0.086, so both models reproduce the demonstrated actions closely. Thus, the closed-loop gap is not explained by open-loop fit on these training windows.

E Matched Comparisons at Five Data Scales

Table 9 lists the five data scales of the matched comparison on RoboTwin 2.0 together with the success rates in both scene conditions. A data scale retains one of every k window starts of the full-data set, after the removal of idle segments, with k equal to 1,000, 100, 20, 10, or 5. At each scale, PatchWAM and the dual-expert control use the same retained windows, the same order of samples, and the same number of training steps; only the random draws of the image augmentation and of the noise differ between the two runs. The number of epochs falls from 100 at 0.1% to 5 at 10% and 20%, so the runs at 1%, 5%, and 10% receive about 42k training steps, whereas those at 0.1% and 20% receive 8.4k and 84k steps. Comparisons across scales therefore combine the effects of data and training length, except among the three scales with about 42k training steps. Because each window spans 16 actions, the retained windows cover a larger fraction of the time steps of the demonstrations than of the window starts: 1.5% at 0.1%, 15% at 1%, 77% at 5%, and at least 97% at 10% and 20%. The windows retained at 1% come from 98% of the 27,225 training demonstrations and those retained at 5% or more from all of them, whereas the 5,369 windows retained at 0.1% come from as many different demonstrations, or 20% of them. Task–condition units whose evaluation stalled in the simulator or exhausted the memory of the renderer were rerun, with the stalled evaluation seeds skipped, so every reported value covers all 100 units.

Table 9: Matched comparisons at five data scales on RoboTwin 2.0 (SR, percent), with 10 episodes per task and condition. Windows are counted after the removal of idle segments, and Steps gives the number of training steps. The first five rows use FLUX.2 [klein] 4B as the backbone, and the last row uses Qwen-Image-2.1.

				Dual-expert control			PatchWAM		
Scale	Windows	Epochs	Steps	Clean	Rand.	Avg.	Clean	Rand.	Avg.
0.1%	5,369	100	8,400	14.8	17.8	16.3	**29.8**	**31.4**	**30.6**
1%	53,683	50	41,950	74.6	75.4	75.0	**88.0**	**88.6**	**88.3**
5%	268,414	10	41,940	77.2	79.6	78.4	**87.6**	**88.4**	**88.0**
10%	536,828	5	41,940	84.4	80.8	82.6	**89.4**	**87.4**	**88.4**
20%	1,073,656	5	83,880	87.2	86.8	87.0	**92.4**	**93.2**	**92.8**
0.1%, Qwen	5,369	100	8,400	23.2	22.2	22.7	**25.8**	**31.0**	**28.4**

Second backbone. To test whether the result holds for another backbone, we replace the FLUX.2 [klein] 4B transformer with the 7B single-stream transformer of Qwen-Image-2.1 (Qwen Team, 2026), together with its autoencoder and text encoder, and keep the windows, number of training steps, batch size, and learning-rate schedule of the 0.1% scale. The dual-expert control on this backbone follows the design of the FLUX.2 control: its randomly initialized action expert has 32 blocks of width 1024, and the block count and the attention dimensions (32 heads of size 128) match those of the backbone. Its action tokens attend to the prefix and to themselves, it adds 0.95B trainable parameters, and at inference it denoises only the action tokens. PatchWAM reaches 28.4% and the control 22.7%, so PatchWAM again succeeds more often, but its gain of 5.7 percentage points is smaller than the 14.3 points with FLUX.2, mainly because the control succeeds more often with Qwen-Image-2.1.

Learning speed at the 5% scale. Figure 3 shows intermediate checkpoints of the two runs at the 5% scale. After 20k training steps, PatchWAM reaches 60.7%, whereas the dual-expert control needs 30k steps to reach a comparable 60.4%. After 30k steps, PatchWAM reaches 81.0% and thus exceeds the final success rate of the control, 78.4% after 41,940 steps. The difference peaks at 37.9 percentage points after 20k steps and narrows to 9.6 points by the end of training. Each curve comes from a single run evaluated with 10 episodes per task and condition, so differences between checkpoints are subject to sampling error.

F Inference Cost

Sharing the transformer between the future frame and the actions makes inference slower. With the same GPU and input for both models, one call that maps an observation to a chunk of 16 actions with 20 solver steps takes 0.446 s for PatchWAM, 1.87 times the 0.239 s of the dual-expert control (mean of ten runs after two warm-up runs). The control runs its visual transformer once, on the instruction and the current frame with the noise level set to zero, and then updates only the 16 action tokens in its smaller action expert at each solver step. PatchWAM instead passes all 721 tokens through the 4B transformer at every solver step, because FLUX.2 modulates every token, including those of the prefix, with the current noise level. The 10-step setting used in the evaluations was not timed. Conditioning the prefix on a fixed noise level would allow it to be cached, and step distillation (Akbari et al., 2026) could reduce the number of solver steps; we have tested neither.

G Additional Analyses

G.1 Untrained Policies in the Matched Comparison

Before training, the dual-expert control already succeeds in 8.7% of the episodes of the matched comparison and PatchWAM in 0.6% (Figure 3). Nine tasks have success conditions that can be triggered by a single contact or a short motion: pressing the stapler, clicking the alarm clock and the bell, turning the switch, opening the microwave and the laptop, moving the playing card away, and the two bottle-shaking tasks. These tasks account for 7.8 of the 8.7 percentage points, because the untrained action expert occasionally satisfies their success conditions by chance.

G.2 Decoupled Noise Levels

PatchWAM applies one noise level to the future-frame and action tokens of a sample (Section 3.3). Three variants built on Self-Flow (Chefer et al., 2026) depart from this choice and are trained and evaluated at the 5% scale of Table 4 (Table 10). The first adopts the dual-timestep scheduling of Self-Flow, which assigns different noise levels to different tokens, together with a representation loss from an exponential-moving-average (EMA) teacher. The second additionally structures the timestep masks by modality, and the third also withholds the action labels of 25% of the samples and replaces them with labels generated by the teacher. Their average success rates are 88.7%, 88.3%, and 85.5%, compared with 88.0% for the shared noise level. The first two variants succeed more often in clean scenes and less often in randomized scenes, and their averages exceed that of PatchWAM by at most 0.7 percentage points, a difference comparable to the uncertainty of a 10-episode evaluation. The third variant reduces the average by 2.5 points. Because every variant

also changes the training objective, these results do not isolate the effect of the noise levels, and the variants are not independent training repetitions of PatchWAM. They give no indication that separate noise levels improve on the shared one in this setting.

Table 10: Self-Flow variants after 41,940 training steps (SR, percent), with one of every 20 window starts retained and 10 episodes per task and condition.

Configuration	Clean	Randomized	Average
Dual-expert control	77.2	79.6	78.4
PatchWAM	87.6	**88.4**	88.0
Self-Flow variant 1	**90.0**	87.4	**88.7**
Self-Flow variant 2	89.4	87.2	88.3
Self-Flow variant 3	85.4	85.6	85.5

H Training Dynamics under the Full-Data Protocol

Figure 5 traces the success rate of PatchWAM over one training run under the full-data protocol. The run uses the clean and randomized demonstrations without the augmented demonstrations and trains for five epochs, or 104,850 training steps with 256 samples per step. The checkpoints at every 10k steps, at 95k steps, and at the end of training are evaluated with 10 episodes per task and condition, and the final checkpoint is also evaluated with 100 episodes. The dashed line marks our 100-episode evaluation of the released ImageWAM checkpoint, which reaches 92.77%, slightly below the 93.38% reported for it in Table 12.

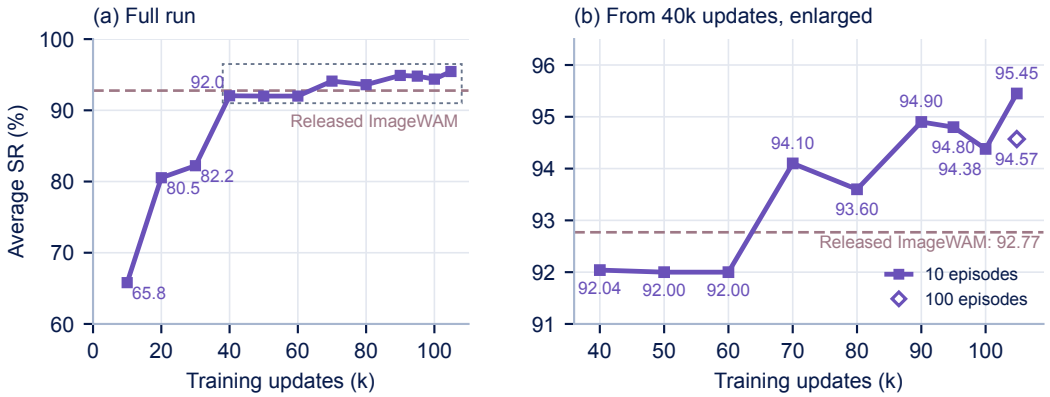


Figure 5: Success rate of PatchWAM over a training run under the full-data protocol without augmented demonstrations. **(a)** The full run of 104,850 training steps. **(b)** The checkpoints from 40k steps onward, enlarged. Squares denote evaluations with 10 episodes per task and condition, the open diamond the 100-episode evaluation of the final checkpoint, and the dashed line the 100-episode evaluation of the released ImageWAM checkpoint.

Most of the improvement occurs within the first two epochs: the success rate rises from 65.8% after 10k training steps to 80.5% after 20k steps and 92.0% after 40k steps. It then stays at 92.0% until 60k steps. From 70k steps onward, every evaluated checkpoint exceeds the released ImageWAM checkpoint, although the intermediate checkpoints are evaluated with fewer episodes. Between 70k steps and the end of training, the success rate varies between 93.60% and 95.45%. Differences of this size between neighboring checkpoints are comparable to the uncertainty of a 10-episode evaluation: the final checkpoint reaches 95.45% with 10 episodes and 94.57% with 100 episodes per task and condition. For the 10-episode value, two of the 100 task–condition units use their 100-episode results because their 10-episode evaluations did not complete. Under the 100-episode evaluation, the final checkpoint exceeds the released ImageWAM checkpoint by 1.80 percentage points. The 96.12% in

Table 11: **Benchmark summary.** PatchWAM against the strongest listed WAM and the strongest listed method in Tables 12–16, in SR (percent) or, for RoboDojo, also the Score. Methods count as WAMs if their authors describe them as such. Values are comparable only within a row; the RoboDojo comparison uses the September 15, 2026 leaderboard snapshot.

			Best listed WAM		Best listed method	
Setting	Metric	PatchWAM	Method	Value	Method	Value
RoboTwin 2.0, full-data	Average	**96.12**	MotuBrain	95.94	MotuBrain	95.94
RoboTwin 2.0, C2R	Average	**79.14**	4D-WAM	61.65	OLA-Sem	71.34
LIBERO	Average	98.4	ABot-M0.5	99.4	QuoVLA	**99.6**
LIBERO-Plus, aug.	Total	**91.8**	ImageWAM	85.3	Hermite-VLAReg	91.4
RoboDojo	Score	**29.39**	OpenWAM-α	17.18	DM0.5	24.90
	SR	**23.60**	OpenWAM-α	11.92	DM0.5	19.34

Table 12 comes from a separate configuration, which adds the 2,100 augmented demonstrations and trains for 110k steps; it exceeds the 94.57% of the run traced in Figure 5 by 1.55 points.

I Full Benchmark Comparisons

Table 11 summarizes the five benchmark settings, and Tables 12–16 list the complete comparisons. The RoboTwin 2.0 clean-to-random baselines follow the official leaderboard (RoboTwin Team, 2026), and the RoboDojo baselines are taken from its leaderboard (RoboDojo Team, 2026).

RoboTwin 2.0, full-data protocol. Both success rates of PatchWAM are the highest among the listed methods, and its two scene conditions differ by only 0.16 percentage points. ImageWAM reaches 93.38%; it uses an image-editing backbone with a separate action expert and provides the codebase on which PatchWAM is built. Appendix H traces the success rate over a full-data training run without the augmented demonstrations.

RoboTwin 2.0, clean-to-random protocol. The average success rate of PatchWAM, 79.14%, exceeds that of the strongest listed baseline, OLA-Sem, by 7.80 percentage points. The drop from clean to randomized scenes is 24.84 percentage points for PatchWAM and between 27.04 and 75.90 points for the other listed WAMs; the strongest listed WAM under this protocol, 4D-WAM, reaches 41.80% in randomized scenes. Under the full-data protocol, which includes randomized demonstrations, the two scene conditions differ by only 0.16 percentage points. This contrast is consistent with the hypothesis that the C2R drop arises mainly from the absence of randomized demonstrations in training. Appendix J breaks these results down by task, and Appendix K shows example rollouts.

LIBERO. Across the four suites, PatchWAM reaches 96.5% on Spatial, 100.0% on Object, 99.0% on Goal, and 98.0% on Long. Long is the weakest suite for 13 of the 14 listed methods, whereas PatchWAM succeeds more often on Long than on Spatial. Only three listed methods exceed its 98.0% on Long, and on Object, PatchWAM matches the best listed result of 100.0%. On Spatial, 11 listed methods succeed more often. Appendix L reports results on LIBERO-Pro.

LIBERO-Plus. The total of 91.8% is reached after 75k training steps on augmented LIBERO-Plus data. PatchWAM also achieves the highest success rates under camera and sensor-noise perturbations, at 97.9% each. The largest differences from ImageWAM arise under camera and robot initial-state perturbations, at 18.1 and 12.3 percentage points, while ImageWAM succeeds more often under language perturbations, at 95.2% compared with 89.1%. Robot initial-state perturbation is the weakest category of PatchWAM, at 71.0%, and five listed methods succeed more often than PatchWAM in it, led by the zero-shot QuoVLA result of 87.6%. These results indicate that robustness to visual perturbations does not necessarily carry over to variations in the initial robot state.

RoboDojo. The Score assigns 100 points to a successful episode and partial credit to an unsuccessful episode according to the intermediate steps completed. Both the Score and the success rate are averaged over the tasks of each capability dimension and then over the five dimensions with

Table 12: **RoboTwin 2.0 success rate (%) under the full-data protocol:** training on clean and randomized demonstrations and evaluation in clean and randomized scenes. Best in **bold** and second-best underlined within each group.

Method	Clean	Randomized	Average
VLA			
ZR-0 (Li et al., 2026b)	88.70	87.98	88.34
HoloBrain-0 (Lin et al., 2026)	91.30	90.80	91.05
HoloBrain-0-QW (Lin et al., 2026)	91.90	92.30	92.10
LingBot-VA (Li et al., 2026d)	92.93	91.55	92.24
AIM (Fan et al., 2026)	94.00	92.10	93.05
G0.5 (Liu et al., 2026)	93.70	92.80	93.25
LingBot-VA 2.0 (Zhang et al., 2026c)	93.80	93.40	93.60
Next Forcing (Xu et al., 2026)	**94.10**	93.50	93.80
Qwen-RobotManip (Yuan et al., 2026a)	93.70	**94.00**	**93.85**
WAM			
X-WAM (Guo et al., 2026)	89.80	90.70	90.25
LaWAM (Chen et al., 2026a)	92.64	89.80	91.22
Faster-WAM (Zhao et al., 2026b)	92.80	92.30	92.55
SelfWAM (Pan et al., 2026)	92.16	93.08	92.62
MECo-WAM (Zhang et al., 2026b)	93.26	91.98	92.62
ImageWAM (Zhang et al., 2026e)	93.20	93.56	93.38
ABot-M0.5 (Chen et al., 2026b)	94.00	94.20	94.10
MotuBrain (MotuBrain Team et al., 2026)	95.80	96.08	95.94
PatchWAM (Ours)	**96.04**	**96.20**	**96.12**

equal weight, so the standard and randomized conditions of the generalization dimension each contribute one tenth (Chen et al., 2026c). The strongest listed WAM in the September 15 leaderboard snapshot, OpenWAM-α, has a Score of 17.18. Under the standard and randomized generalization conditions, PatchWAM succeeds in 33.33% and 16.33% of the episodes, compared with at most 28.00% and 6.00% for the other listed methods. It reaches 23.00% on precise manipulation, compared with at most 20.42%, and 22.00% in open-vocabulary instruction following, compared with at most 4.25%. On long-horizon tasks, it reaches 31.50%, below the 32.25% of G0.5, and on memory tasks it reaches 16.67%, compared with 47.44% for DM0.5.

Table 13: **RoboTwin 2.0 success rate (%) under the clean-to-random protocol:** training on clean demonstrations only and evaluation in clean and randomized scenes. Best in **bold** and second-best underlined within each group.

Method	Clean	Randomized	Average
VLA			
StarVLA (StarVLA Community, 2026)	46.52	3.16	24.84
GalaxeaVLA (Jiang et al., 2025)	62.70	12.72	37.71
Xiaomi Robotics-0 (Cai et al., 2026b)	62.90	18.20	40.55
EventVLA (Yang et al., 2026a)	65.60	15.70	40.65
Spatial Forcing (Li et al., 2026a)	77.20	26.74	51.97
OLA-Geo (RoboTwin Team, 2026)	**82.96**	33.64	58.30
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	70.70	46.00	58.35
GigaBrain-0.7 (GigaBrain Team et al., 2026)	66.80	**67.90**	67.35
OLA-Sem (RoboTwin Team, 2026)	75.12	67.56	**71.34**
WAM			
AHA-WAM (Cai et al., 2026a)	64.30	3.20	33.75
FastWAM (Yuan et al., 2026b)	77.80	1.90	39.85
ABot-M0 (Yang et al., 2026c)	57.40	30.36	43.88
X-WAM (Guo et al., 2026)	70.00	25.80	47.90
4D-WAM (Yang et al., 2026b)	81.50	41.80	61.65
PatchWAM (Ours)	**91.56**	**66.72**	**79.14**

Table 14: **LIBERO success rate (%) on the four task suites.** Best in **bold** and second-best underlined within each group.

Method	Spatial	Object	Goal	Long	Average
VLA					
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5
WorldVLA (Cen et al., 2025)	87.6	96.2	83.4	60.0	81.8
UniVLA (Wang et al., 2026d)	95.4	98.8	93.6	94.0	95.4
F1-VLA (Lv et al., 2025)	98.2	97.8	95.4	91.3	95.7
OpenVLA-OFT (Kim et al., 2025)	97.6	98.4	97.9	94.5	97.1
TurboVLA (Xie et al., 2026a)	99.2	99.8	97.4	94.2	97.6
StarVLA-α (Specialist) (Ye et al., 2026b)	98.7	99.7	98.6	94.2	97.8
X-VLA (Zheng et al., 2026)	98.2	98.6	97.8	97.6	98.1
S^2-VLA (Xie et al., 2026b)	98.4	99.6	98.4	96.4	98.2
FocusVLA (Zhang et al., 2026d)	99.6	**100.0**	98.8	96.2	98.6
GeoAlign (Chen et al., 2026d)	**100.0**	99.6	99.8	96.6	99.0
CORAL (SimVLA) (Luo et al., 2026c;d)	99.6	99.8	99.0	**98.8**	99.3
QuoVLA (Wang et al., 2026c)	99.8	99.9	**100.0**	98.7	**99.6**
WAM					
ABot-M0.5 (Chen et al., 2026b)	**100.0**	99.8	**99.4**	**98.4**	**99.4**
PatchWAM (Ours)	96.5	**100.0**	99.0	98.0	98.4

Table 15: **LIBERO-Plus success rate by perturbation category.** Best in **bold** and second-best underlined within each group.

Method	Camera	Robot	Language	Light	Background	Noise	Layout	Total
VLA								
π_0 (Black et al., 2024)	79.6	21.1	72.5	84.7	86.2	68.3	69.4	68.8
OpenVLA*-Full (Kim et al., 2024)	69.4	49.6	66.3	88.2	88.5	78.7	70.3	73.0
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	70.3	41.7	81.1	97.3	94.6	71.8	84.9	77.4
InternVL-3.5 + DiT (Wang et al., 2025)	94.2	32.4	64.5	94.7	93.0	94.7	74.8	78.3
ELAN4D ($\pi_{0.5}$) (He et al., 2026)	63.7	70.7	77.8	89.8	91.4	79.9	81.4	79.2
GR00T-N1.6 (NVIDIA GEAR Team et al., 2025)	92.6	33.5	80.1	93.6	95.4	93.6	75.0	80.5
OpenVLA-OFT+ (Kim et al., 2025)	92.8	30.3	85.8	94.9	93.9	89.3	77.6	80.7
SRPO (Fei et al., 2026a)	83.4	62.0	73.6	97.2	97.7	85.7	75.2	82.1
RoVLA (Luo et al., 2026b)	**96.6**	32.0	**91.5**	95.9	96.1	95.1	74.1	83.0
$\pi_{0.5}$ + AXIS (Zhao et al., 2026a)	83.8	78.2	88.3	96.5	98.1	96.2	85.5	89.5
CAC-VLA (Xiong et al., 2026)	91.2	78.4	83.3	97.5	97.1	95.4	87.8	90.1
QuoVLA (Wang et al., 2026c)	82.3	**87.6**	88.2	98.2	95.9	90.8	89.3	90.3
Anchor-Align VLA (Dalal et al., 2026)	96.3	59.1	87.2	**99.0**	**99.6**	**96.9**	**97.4**	90.8
Hermite-VLAReg (Lv et al., 2026)	89.2	85.4	87.3	97.7	96.7	93.8	89.9	**91.4**
WAM								
Fast-WAM (Yuan et al., 2026b)	16.4	44.5	68.9	78.2	53.7	37.7	60.7	50.0
Being-H0.7 (Luo et al., 2026a)	-	-	-	-	-	-	-	82.1
Cosmos-Policy (Kim et al., 2026)	75.8	63.3	81.7	96.5	88.9	92.7	82.2	82.2
ImageWAM (Zhang et al., 2026e)	79.8	58.7	**95.2**	96.1	91.2	93.3	83.1	85.3
ABot-M0.5 (Chen et al., 2026b)	70.5	**87.4**	88.6	94.0	89.7	75.5	85.2	83.4
PatchWAM	**97.9**	71.0	89.1	**98.5**	**96.9**	**97.9**	**92.3**	**91.8**

Table 16: **RoboDojo simulation results across capability dimensions**, with the Score and the success rate (SR, percent) of each dimension. Gen-Std and Gen-Rand denote generalization under standard and randomized conditions, Precision precise manipulation, Long long-horizon tasks, and Open open-vocabulary instruction following. Baselines are the ten strongest entries of the leaderboard on September 15, 2026. Best in **bold** and second-best underlined within each group.

	Gen-Std		Gen-Rand		Precision		Long		Memory		Open		Average	
Method	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR
VLA														
StarVLA-PI_v3 (StarVLA Community, 2026)	19.08	14.44	3.36	1.67	17.77	12.50	18.46	11.00	4.59	4.00	2.03	2.00	10.81	7.51
InternVLA-A1.5 (Ma et al., 2026)	16.81	11.78	3.90	1.89	15.23	10.17	23.80	13.75	4.93	3.56	1.43	1.42	11.15	7.14
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	20.93	14.89	5.82	1.44	12.40	5.50	23.54	14.67	5.78	4.56	1.98	1.67	11.41	6.91
Spatial Forcing (Li et al., 2026a)	21.25	14.89	6.98	3.78	17.32	10.58	23.26	14.58	5.43	4.11	1.78	1.58	12.38	8.04
Hy-Embodied-0.5-VLA (Zhang et al., 2026a)	21.98	16.56	1.57	0.22	13.81	8.00	25.74	14.92	13.37	12.11	0.65	0.58	13.07	8.80
Meituan-Robotics-0 (RoboDojo Team, 2026)	19.28	12.67	8.22	3.67	16.77	7.75	29.61	18.58	10.06	8.89	**4.54**	**4.25**	14.95	9.53
Xiaomi-Robotics-1 (Xiaomi Robotics Team et al., 2026)	**35.65**	**28.00**	**11.44**	**6.00**	26.69	18.83	38.39	23.67	7.81	6.56	3.94	3.58	20.07	13.93
GalaxeaVLA (G0.5) (Liu et al., 2026)	27.72	21.44	9.20	4.22	**28.25**	**20.42**	**44.12**	**32.25**	8.61	7.33	1.73	1.58	20.23	14.88
DM0.5 (Dexmal Team, 2026)	23.49	17.89	8.06	4.00	24.82	16.75	33.70	19.50	**47.74**	**47.44**	2.43	2.08	**24.90**	**19.34**
WAM														
OpenWAM-α (Wang et al., 2026e)	33.16	25.56	8.26	4.11	18.45	9.25	34.93	25.33	10.41	9.11	1.41	1.08	17.18	11.92
PatchWAM (Ours)	**39.08**	**33.33**	**22.97**	**16.33**	**30.96**	**23.00**	**43.75**	**31.50**	**18.48**	**16.67**	**22.74**	**22.00**	**29.39**	**23.60**

J PER-TASK RESULTS UNDER THE CLEAN-TO-RANDOM PROTOCOL

Figure 6 breaks down the clean-to-random results of Table 13 by task. It compares PatchWAM at the reported checkpoint with the four strongest baselines of that table, whose per-task results are taken from the leaderboard (RoboTwin Team, 2026); every method is evaluated with 100 episodes per task and condition. In clean scenes, PatchWAM succeeds in every episode of 11 tasks and falls below 80% on five tasks, the two lowest being putting the object into the cabinet at 48% and hanging the mug at 51%. In randomized scenes, it reaches at least 80% on 16 tasks and at most 20% on two, rotating the QR code and moving the stapler to the pad. Compared with OLA-Sem and GigaBrain-0.7, whose success rates in randomized scenes are close to its own, PatchWAM succeeds more often in randomized scenes on 23 and 24 tasks, respectively, and less often on 26 tasks each, so policies with similar success rates differ in which tasks they solve.

	(a) Clean scenes					(b) Randomized scenes				
	PatchWAM	OLA-Sem	GigaBrain-0.7	4D-WAM	$\pi_{0.5}$	PatchWAM	OLA-Sem	GigaBrain-0.7	4D-WAM	$\pi_{0.5}$
stack blocks two	100	91	80	100	91	97	90	92	34	43
move playingcard away	100	90	58	82	84	96	92	82	35	65
stack blocks three	98	74	60	95	77	96	52	78	82	25
place burger fries	97	94	82	86	82	95	75	86	72	86
place cans plasticbox	98	97	8	94	31	93	85	18	48	73
blocks ranking rgb	95	64	68	88	72	92	66	76	10	44
grab roller	100	100	100	100	99	92	100	100	68	97
shake bottle horizontally	100	92	100	98	99	91	70	92	70	100
stack bowls two	98	94	90	94	94	91	90	74	81	70
place object stand	98	81	88	87	85	87	77	76	42	47
shake bottle	100	89	100	98	100	85	70	98	72	100
pick dual bottles	100	92	26	92	72	84	84	56	67	57
place empty cup	99	98	90	99	96	83	94	90	70	80
place bread basket	93	84	60	87	63	82	80	68	77	60
place container plate	99	97	94	93	92	80	92	94	82	72
put bottles dustbin	88	44	68	68	78	80	37	54	19	50
click alarmclock	100	100	98	100	65	77	91	96	70	50
place shoe	94	79	88	83	82	76	78	92	52	50
dump bin bigbin	98	96	90	86	92	75	94	66	57	84
place dual shoes	94	87	68	66	70	74	76	72	19	45
place object basket	91	54	74	75	70	74	49	64	40	29
adjust bottle	100	72	98	80	99	73	71	98	36	77
handover block	96	24	16	79	30	73	17	32	25	12
place bread skillet	89	81	80	82	76	73	67	50	69	47
place object scale	99	69	64	79	78	73	63	54	24	38
stack bowls three	86	88	70	82	81	72	81	64	27	52
click bell	100	100	98	98	31	71	98	84	77	35
press stapler	91	98	90	76	79	71	95	74	61	63
scan object	87	42	32	73	49	71	32	46	53	40
lift pot	99	100	86	88	98	69	91	86	9	35
pick diverse bottles	80	76	38	77	66	67	77	50	44	42
open laptop	96	90	78	85	93	65	74	88	82	68
place a2b left	100	65	68	71	71	65	64	72	19	49
handover mic	99	89	100	85	98	61	80	100	13	6
place a2b right	94	85	58	62	68	58	80	68	42	39
move pillbottle pad	99	78	32	89	46	56	71	58	22	32
hanging mug	51	16	26	70	17	55	15	36	10	14
beat block hammer	100	65	94	92	93	51	55	74	28	23
place can basket	74	36	20	83	56	51	34	22	51	10
blocks ranking size	69	36	42	54	45	50	37	28	8	21
place phone stand	95	75	36	84	65	46	63	52	21	37
place mouse pad	89	59	26	54	34	45	55	44	9	31
open microwave	88	76	82	68	86	37	43	50	12	59
stamp seal	94	87	52	74	53	37	70	54	10	21
turn switch	67	86	42	86	52	36	70	52	11	25
place fan	87	76	78	48	66	32	68	74	17	26
put object cabinet	48	40	78	69	39	28	27	82	47	22
move can pot	95	46	86	94	60	25	45	82	17	10
move stapler pad	82	58	26	43	20	15	53	40	12	18
rotate qrcode	84	46	54	79	90	10	40	58	67	21

Figure 6: Per-task success rates (percent) under the clean-to-random protocol for PatchWAM at the reported checkpoint and the four strongest baselines of Table 13, all evaluated with 100 episodes per task and condition. Tasks are sorted by the success rate of PatchWAM in randomized scenes. **(a)** Clean scenes. **(b)** Randomized scenes.

K Qualitative Clean-to-Random Rollouts

Figure 7 shows paired outcomes from four difficult tasks in randomized scenes. The examples are selected from all 100 evaluation episodes per task of the clean-to-random PatchWAM checkpoint after 100k training steps. Success and failure labels are produced by the RoboTwin evaluator rather than assigned manually.



Figure 7: **Qualitative clean-to-random rollouts in randomized scenes.** Each row pairs a successful rollout (green) with a failed rollout (red) from the same task, with four frames ordered from early interaction to the final outcome. The four tasks have success rates of 4%, 9%, 25%, and 48% in the evaluation of this checkpoint, so each successful rollout comes from a task that the model completes in at most half of the episodes.

L LIBERO-Pro Comparison

LIBERO-Pro (Zhou et al., 2025) evaluates policies trained on the original LIBERO data under changes to the objects, to their initial positions, to the wording of the instruction, and to the task itself. Table 17 gives the success rate under each perturbation and on the unperturbed tasks, averaged over the four LIBERO suites. The results of π_0, OpenVLA, and $\pi_{0.5}$ are taken from Zhou et al. (2025), and those of SimVLA and QuoVLA from their papers.

PatchWAM reaches an average success rate of 57.9%, above π_0 and OpenVLA and below the other three listed methods. On the unperturbed tasks and under paraphrased instructions, it succeeds in 98.4% and 96.8% of the trials, 1.4 and 2.7 percentage points below the best listed results. Under object perturbations, its success rate of 80.8% is the lowest among the listed methods. Changes to the initial positions and to the task are difficult for every listed method: PatchWAM reaches 8.8% and 5.0%, and only QuoVLA exceeds 21% in either category.

Table 17: LIBERO-Pro results by perturbation (SR, percent), averaged over the four LIBERO suites. Original denotes the unperturbed tasks. Methods are sorted by average SR, and PatchWAM is shown in bold; averages are computed before rounding.

Method	Original	Object	Position	Semantic	Task	**Average**
π_0 (Black et al., 2024)	92.3	90.5	0.0	90.5	0.0	54.7
OpenVLA (Kim et al., 2024)	97.0	93.0	0.0	97.3	0.0	57.5
SimVLA (Luo et al., 2026d)	98.5	81.5	8.3	98.8	6.0	58.6
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	96.5	96.0	20.8	95.8	0.8	62.0
QuoVLA (Wang et al., 2026c)	99.8	88.3	36.0	99.5	25.5	69.8
PatchWAM (Ours)	**98.4**	**80.8**	**8.8**	**96.8**	**5.0**	**57.9**

M Extended Discussion of Limitations

Scope of the architectural evidence. The matched comparisons show that a separate action expert is not required at the five data scales we tested, but they do not establish that every dedicated action head is unnecessary. Relative to the dual-expert control, PatchWAM changes action-specific capacity, token visibility, and the processing path together. The controls of Section 4.3 separate two of these factors under the full-data protocol: a learned linear interface performs on par with the fixed codec, and removing the visibility between action and future-frame tokens costs 0.58 percentage points. Because they were run in a different data regime, they do not decompose the gaps of the matched comparisons, and no comparison in this paper includes independent training seeds. The interventions of Appendix D show that the predicted future frame informs action selection in the full-data model, but the isolated-attention control shows that the success rate depends little on this link in that regime. The probes are linear and use windows of the training data, on which the predicted future frames are accurate; they show that the predicted future carries the action's motion, not that every task relies on it. At the highest noise level, the future-frame tokens contain no information about the observed future, and a single future frame can be consistent with several action sequences. Additional conditioning information can reduce the squared error of an ideal predictor, although this does not imply higher closed-loop success for a trained model. The future frame shows only the end of a chunk, so it constrains the earlier actions only together with the current state. The action equals the next joint state exactly on RoboTwin 2.0 and RoboDojo but is an end-effector displacement on LIBERO, where the two readouts are related only approximately. Generating the action with the pretrained field is therefore supported by the matched comparisons, the probes, and the interventions, but no single experiment isolates it from the other differences to the dual-expert control.

Inference and deployment. Because future-frame and action tokens are updated at every solver step, one inference call with 20 solver steps takes 1.87 times as long as in the dual-expert control. The 10-step setting used for evaluation was not timed. Part of this cost follows from the architecture, which passes the future-frame tokens through the full transformer at every solver step. In addition, unlike the dual-expert control, PatchWAM recomputes the conditioning prefix at every solver step, because FLUX.2 modulates the prefix with the current noise level (Appendix F). All experiments are conducted in simulation, and the results do not establish real-time operation, performance on physical robots, or safe deployment.