Extended Abstract

Motivation Many robot policies now improve by predicting a generated future—an image, video rollout, latent forecast, or language sub-goal—before acting. The unresolved question is *why* this helps. A future-prediction loss might simply align the shared representation, making the generator a training-time scaffold that can be dropped at deployment. Or the action policy may need to read the generated future as an explicit target-state cue, making the generator part of the inference-time policy. These mechanisms imply opposite engineering choices, but prior systems usually entangle them.

Method We isolate the two mechanisms inside a single World-Action Model instead of comparing different architectures. The model is a shared flow-matching diffusion transformer that denoises future-video latents and action chunks over short autoregressive blocks. We expose two binary switches: whether the video loss back-propagates through the shared backbone, testing *representation alignment*, and whether action tokens may attend to generated-video tokens, testing *explicit conditioning*. Turning the switches on additively gives an *action-only* baseline, a *representation-alignment* variant that predicts video but prevents the action from reading it, and *joint denoising*, where the action attends to the co-generated future. We also test *self-feedback*, which feeds a finalized generated frame from one control step back into the next.

Implementation We adapt Wan2.2-TI2V-5B into a joint video–action policy and full-fine-tune it on LIBERO. Each clip contains two side-by-side camera views at 160×320, language, proprioception, and 24-step chunks of 7-D end-effector delta actions. Training uses 1693 teleoperation demonstrations ($\sim$273k frames), bf16 AdamW, and DeepSpeed ZeRO-2 on seven H100 training GPUs. Evaluation runs in LIBERO/MuJoCo on four suites of ten tasks each. Besides binary success, we report a simulator-state progress metric that gives ordered partial credit for reaching, grasping, transporting, and placing objects.

Results The ablation gives a clear answer. At the matched 40k-step budget, the pure action policy is at the floor: 0.0% success and 9.4% progress. Adding future prediction only as representation alignment barely moves it, reaching 1.7% success and 15.9% progress, even though the same 5B backbone is now trained to predict video. Letting the action read the generated future changes the regime: joint denoising reaches 27.5% success and 55.7% progress, while self-feedback reaches 32.5% success and 60.6% progress. In additive terms, alignment contributes only $+6.5$ progress points over the action-only floor, whereas conditioning contributes a further $+39.8$ points; for success, conditioning contributes roughly $15\times$ as much as alignment. The stage analysis shows where the gain appears: non-conditioning policies often reach toward objects but rarely grasp them (3.1% grasp rate for action-only and 15.5% for alignment), while explicit conditioning raises grasping to 62.8% for joint denoissing and 55.8% for self-feedback.

Discussion In this backbone and benchmark, generated futures help primarily as inputs, not as representation-shaping regularizers. Reading noisy video latents during joint denoising and reading a finalized frame through cross-step feedback perform similarly, suggesting that the precise interface matters less than whether the action pathway receives the forecast at all. The deployment implication is direct: keeping only the cheap action path would discard nearly all of the benefit. The generator should instead remain coupled to control and be optimized for faster, more accurate inference.

Conclusion Generated futures help this World-Action Model on LIBERO because the policy conditions on them, not because video prediction alone aligns the representation. The next step is to test whether the same mechanism holds across other robot-policy backbones, embodiments, benchmarks, and future modalities.

Do generated futures help robot policies through representation alignment or explicit conditioning?

Minyeong Kim
Department of Computer Science
Stanford University
minyeong@stanford.edu

Abstract

Predicting a generated future—a future image, video, or language sub-goal—is a popular ingredient in modern robot policies and reliably improves performance, yet why it helps is rarely isolated. Two mechanisms are usually entangled: *representation alignment*, where the future-prediction objective shapes the policy's internal representation during training, and *explicit conditioning*, where the generated future is fed into the action path as input. We disentangle them in a World-Action Model built on a video diffusion backbone: a shared DiT that denoises an action chunk together with an optional future-video stream. Starting from a pure action policy baseline, we add representation alignment by co-training with a video loss, and explicit conditioning in two forms: *joint denoising*, where the action attends to the generated video latents, and *self-feedback*, where the model feeds its own generated frame back as input at the next step. On four LIBERO suites, representation alignment alone yields a negligible lift over the action-only baseline, whereas both forms of explicit conditioning account for almost the entire gain. Our results suggest that, in this setting, generated futures help primarily as *input* rather than as a representation-shaping regularizer.

1 Introduction

A growing number of robot-learning systems pair an action policy with a model that predicts a generated future: a future image, a short video rollout, a latent forecast, or a language sub-goal. Such systems repeatedly report gains over policies that map observations directly to actions [2, 4, 5, 19, 20]. The empirical case for "predict the future" is by now strong.

What is far less clear is *why* predicting the future helps. At least two mechanisms are plausible. The first is **representation alignment**: training the model to predict the future is an auxiliary objective that shapes the shared backbone, so the representation used by the action head becomes more predictive and structured even if the forecast itself is never read at deployment. The second is **explicit conditioning**: the generated future is fed into the action computation as additional input, so that the action head effectively learns an inverse dynamics model—inferring the action that carries the system from the current observation to the generated future. Because the two mechanisms are typically confounded within a single model, a reported gain cannot be attributed to either one in isolation.

We disentangle the two mechanisms with a controlled study on a single World-Action Model built on a video diffusion backbone. At its core, a diffusion transformer (DiT) [11] denoises an action chunk at each step by flow matching [7], running autoregressively over short blocks of observation frames. Onto this action path we introduce two minimal switches, each isolating one mechanism. The first switch governs representation alignment: it determines whether a dynamics loss, a flow-matching objective on the predicted future video, back-propagates into the shared backbone. The second switch governs explicit conditioning: it determines whether the action tokens may attend to the generated video, letting the action condition on the forecast.

We then sweep these switches additively. Our baseline is a pure action policy with both switches off, so the model neither predicts the future nor reads it. We first add representation alignment by enabling the dynamics loss, which co-trains the backbone to predict future video while the action pathway still cannot attend to it. We then add explicit conditioning, which we study in two forms. Under *joint denoising*, the action attends to latent video tokens co-generated within the same forward pass. Under *self-feedback*, the model instead consumes its own output over time: during inference, the frame it generates at one step is fed back as input at the next. With the data, backbone, and optimizer

held fixed throughout, the change in performance from one configuration to another isolates the contribution of the mechanism we have just added.

We evaluate on four LIBERO manipulation suites [8]—SPATIAL, OBJECT, GOAL, and LONG—and report two metrics: the binary success rate and a finer-grained, stage-decomposed progress score that awards partial credit for reaching the intermediate stages of a task. Comparing all configurations at a matched budget of 40k training steps, the macro-averaged results give a clear answer. The pure action policy essentially fails, reaching 0% success and only 9.4% progress. Adding representation alignment alone barely lifts this floor, to 1.7% success and 15.9% progress, even though the backbone is now trained to predict the future. Explicit conditioning, by contrast, supplies almost the entire gain: joint denoising reaches 27.5% success and 55.7% progress, and self-feedback reaches 32.5% and 60.6%, raising the success rate more than an order of magnitude above the alignment-only model. The two conditioning variants perform comparably, suggesting the benefit comes from the action reading the generated future rather than from the particular way it is supplied. Because compute limited training to 40k steps—short of saturation in both metrics—these figures are conservative and would likely improve with longer training.

Our contributions are as follows:

- **A clean decomposition.** We isolate representation alignment from explicit conditioning within a single World-Action Model—the former via the dynamics-loss gradient into the shared backbone, the latter via the action's attention to the generated video—holding data, backbone, optimizer, and evaluation fixed so each comparison varies exactly one mechanism.
- **An empirical answer on LIBERO.** Across four suites, representation alignment alone barely beats a pure action policy, whereas both forms of explicit conditioning—joint denoising and self-feedback—account for almost all of the gain; in a shared-DiT World-Action Model, the generated future is therefore used primarily as an input.
- **Future work.** Promising directions include other forms of generated future (e.g., predicted language sub-goals in VLA models) and evaluation across more benchmarks and embodiments, toward a more general training recipe for action models.

2 Related Work

World Action Models and video-based robot policies. Generative robot policies have increasingly moved from directly regressing actions to denoising structured action chunks conditioned on vision and language. Diffusion Policy [3] is a canonical action-only version of this idea, while recent robot foundation policies scale diffusion or flow objectives to larger, language-conditioned action models [1, 9]. The line most closely related to our work adds visual prediction to the action model: GR-1 [19] predicts future images and actions after video pretraining, PAD [5] performs future-image prediction and action denoising in a shared DiT, DreamZero [20] formulates a World Action Model that jointly models video and actions with a pretrained video diffusion backbone, and newer video-generator policies [6, 15] similarly pair imagined visual rollouts with action extraction or joint action prediction. These systems motivate the premise that predicting the future is useful for control, but they usually introduce the auxiliary video objective and an action-visible future pathway together. A performance gain therefore does not reveal whether the future helped by shaping the shared representation or by serving as an input to the action path.

Generated futures in robot policies. Another branch treats futures as explicit goals or plans outside the low-level action model. UniPi [4] casts sequential decision making as text-conditioned video generation followed by action extraction, and SuSIE [2] uses image-editing diffusion to propose intermediate visual subgoals for a goal-conditioned controller. These approaches make the "future as input" role explicit, whereas auxiliary prediction objectives in joint policies leave open the possibility that the benefit is only representation learning. Our work is complementary: rather than proposing a new generator or benchmark, we take one video–action architecture and toggle exactly two interfaces—the video-loss gradient into the backbone and the action's attention/feedback access to generated video. This controlled sweep targets the mechanism that prior work leaves entangled.

3 Method

Architecture

We study a World-Action Model: a single network that, at every control step, both imagines what the robot will see next and decides how to act. It is built on a pretrained video diffusion model whose core is a diffusion transformer, hereafter DiT, originally trained to generate video [11, 18]. Figure 1 shows these components. The architecture is largely inspired by DreamZero [20].

Every input is mapped to tokens that the shared DiT processes as one sequence. The current camera observation is compressed by a VAE encoder into a clean context token, written c or `ctx`, that represents the present. Video

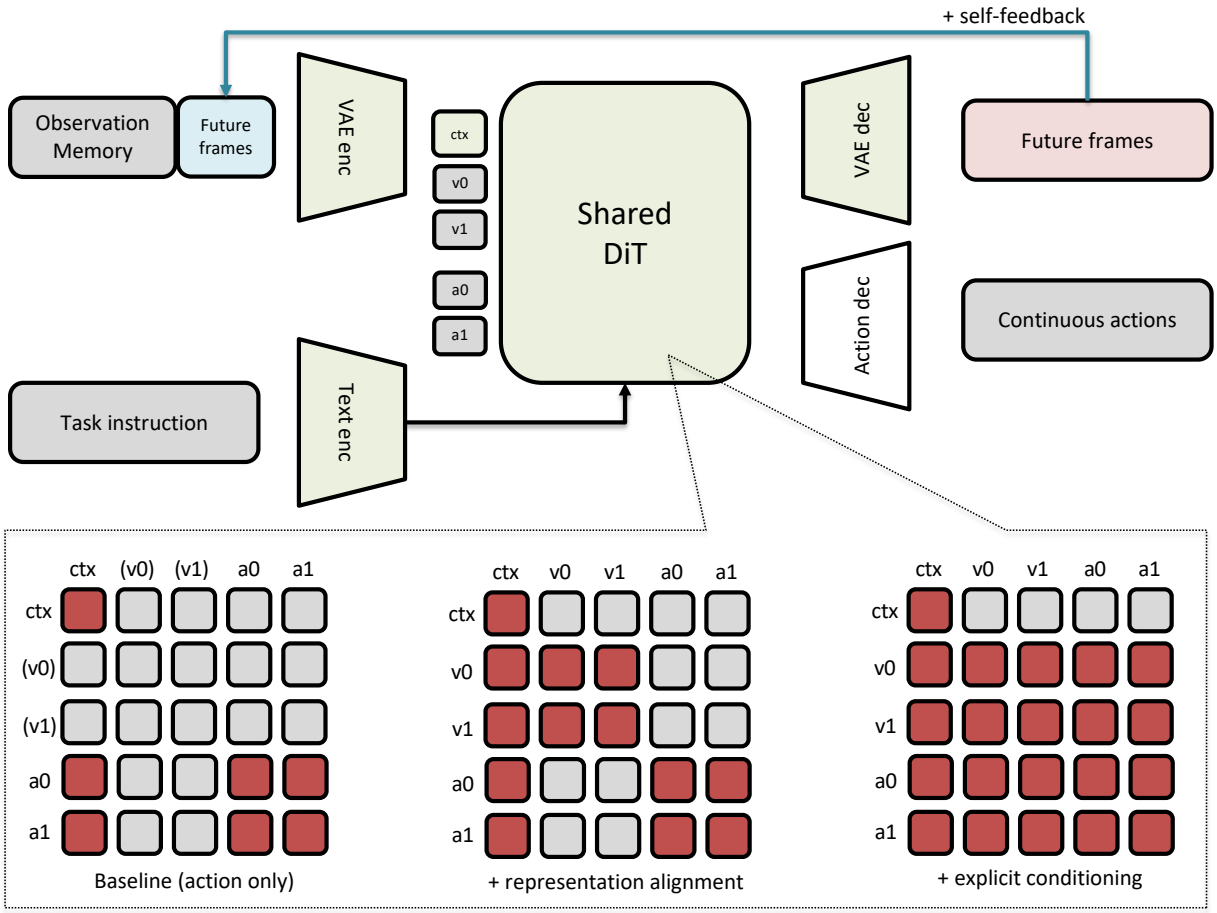


Figure 1: **Architecture and attention masks.** *Top:* the shared DiT processes a context token `ctx`, video tokens v_i, and action tokens a_i, with the task instruction supplied through a text encoder; VAE and action decoders emit the future frames and continuous actions, and self-feedback re-feeds a generated frame at the next step. *Bottom:* attention masks for the three variants, where a filled cell means the row token attends to the column token, and `ctx` is visible to every token. The action attends to the video only under explicit conditioning.

tokens $v_0, v_1, \ldots$ stand for the future frames the model imagines, and during generation they begin as noise and are progressively denoised. Action tokens $a_0, a_1, \ldots$ are the latent form of the upcoming action chunk and are likewise denoised from noise. The task instruction ℓ is embedded by a text encoder and supplied to every layer, and the proprioceptive state s is provided alongside the action tokens. After the DiT, a VAE decoder turns denoised video tokens back into pixel frames and an action decoder turns denoised action tokens into continuous robot commands.

Generation proceeds one short block of frames and actions at a time. It is autoregressive in time but operates over a bounded context: rather than attending to the entire history, each block attends through a fixed-length causal window over the most recent frames, so the per-step cost stays constant along a rollout. During training the model denoises a full clip at once, with each block attending to the clean ground-truth frames of the earlier blocks under teacher forcing; at inference the context frames instead come from the real observations the policy receives as it runs.

Training objective

Both the video and the actions are produced by flow matching [7], a diffusion-style objective that learns to carry Gaussian noise to data along a straight line. Consider a target x, either a block of video latents or an action chunk, together with its conditioning c. Let x_1 be the ground truth and $x_0 \sim \mathcal{N}(0, I)$ be noise. The interpolant $x_\tau = (1-\tau)\, x_0 + \tau\, x_1$ for $\tau \in [0, 1]$ moves at the constant velocity $x_1 - x_0$, and the network f_θ is trained to predict that velocity:

$$\mathcal{L}(x \mid c) = \mathbb{E}_{\tau \sim \mathcal{U}[0,1],\ x_0 \sim \mathcal{N}(0,I)} \left\| f_\theta(x_\tau, \tau, c) - (x_1 - x_0) \right\|_2^2. \tag{1}$$

At inference one starts from noise and integrates $\dot{x} = f_\theta(x, \tau, c)$ from $\tau{=}0$ to $\tau{=}1$ to obtain a sample. Applying Eq. (1) to the two targets gives a dynamics loss on the video and an action loss,

$$\mathcal{L}_{\text{dyn}} = \mathcal{L}(v \mid c_v)\,, \qquad \mathcal{L}_{\text{act}} = \mathcal{L}(a \mid c_a)\,, \tag{2}$$

where c_v and c_a denote the tokens the video and the action respectively attend to. Each variant sets its own c_v and c_a, specified below. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda \mathcal{L}_{\text{dyn}}, \qquad \lambda \in \{0, 1\}. \tag{3}$$

Two quantities control the mechanisms we study. The coefficient λ in Eq. (3) decides whether predicting the future reshapes the shared backbone, which is **representation alignment**. Whether the action's context c_a includes the video, coupling the two pathways so the action can read the forecast, is **explicit conditioning**.

Experiment Design

We design the variants to isolate one mechanism at a time, building them additively from a policy that uses neither: turning on the dynamics loss adds representation alignment, and coupling the two pathways by attention adds explicit conditioning, which we study in two forms. Figure 1 shows the corresponding attention masks.

Action-only baseline. We set $\lambda = 0$ and $c_a = \{c, \ell, s\}$. Only the action loss carries a gradient, and the action tokens attend to the context, the instruction, the state, and one another, never to the video. Training minimizes $\mathcal{L} = \mathcal{L}_{\text{act}}$. At inference the model encodes the current observation into c, denoises an action chunk from noise given c_a, and executes it, generating no future frames.

+ Representation alignment. We turn on the dynamics loss, $\lambda = 1$, but keep the video and action pathways fully decoupled: neither attends to the other, so each conditions only on the observation context, the instruction, and its own state, $c_v = c_a = \{c, \ell, s\}$. The future-prediction objective therefore helps shape the shared backbone while the action still cannot read the generated future. Training minimizes $\mathcal{L} = \mathcal{L}_{\text{act}} + \mathcal{L}_{\text{dyn}}$. Inference is identical to the baseline: the action is denoised from c_a, and the video need not be generated at all, since the action does not read it.

+ Explicit conditioning. We now let the action read the generated future, so the action head behaves like an inverse dynamics model that infers the action from the current observation and a predicted future. We study two forms that differ in how the future is supplied to the action.

Joint denoising. The video and the action describe the same timestep and are denoised together. We restore the attention coupling between the two pathways so they attend to each other, $c_a = \{c, \ell, s, v\}$ and $c_v = \{c, \ell, a, s\}$; in one forward pass the action reads the video while it is still being generated, conditioning on the model's own forecast. Training minimizes $\mathcal{L} = \mathcal{L}_{\text{act}} + \mathcal{L}_{\text{dyn}}$, the same objective as representation alignment but with the cross-attention restored. At inference each step jointly produces the next video block and action chunk, and the action reads the still-denoising video latents.

Self-feedback. This form recycles the model's own forecast across control steps. We shift the video target one block ahead of the action, so the model predicts the frame that follows the action. At inference, the block the model generates at one step is concatenated with the next step's real observation as an extra clean context frame, so the action conditions on its own forecast alongside the real observation. To keep this stable, the model rebuilds its context around the fresh real observation every step and carries forward only the single most-recent generated block, so the rollout never free-runs on an accumulating stack of its own predictions. Training adds the one-block shift to the joint objective $\mathcal{L} = \mathcal{L}_{\text{act}} + \mathcal{L}_{\text{dyn}}$.

4 Experimental Setup

Backbone and policy. We build on a 5B-parameter video diffusion backbone (Wan2.2-TI2V-5B [17, 18]; hidden size 3072, 30 layers, a 48-channel latent VAE) and adapt it into a joint video–action model that we full-fine-tune (no LoRA). The policy reads two camera views, an agent view and a wrist camera, placed side by side and rendered at 160×320. Its proprioceptive input is an 8-dimensional end-effector state—3 for position, 3 for axis-angle orientation, and 2 for the gripper fingers—while its action is a 7-dimensional command, a 6-DoF end-effector delta plus a single gripper open/close; the gripper accounts for the mismatch, since two finger positions are observed but only one command is emitted. Each training clip is 33 frames, which the VAE compresses to 9 latent frames: one leading context frame followed by four blocks of two latent frames each. Every block pairs its two video frames with one proprioceptive-state token and one action chunk of 24 actions (the action horizon).

Benchmark. We use LIBERO [8], a suite of language-conditioned tabletop manipulation tasks. Training uses its 1693 teleoperation demonstrations ($\sim$273k frames), which we re-encode to on-disk MP4 for the video loader; actions are stored as end-effector deltas and used directly, and we recompute dataset statistics for quantile (q01/q99) normalization. We evaluate in the LIBERO MuJoCo simulator [16] on four task suites—SPATIAL, OBJECT, GOAL, and LONG—of ten tasks each, and in our main evaluation we run every task for ten trials from different initial states. We report two

Table 1: **Main results on LIBERO** (success rate / progress, %; mean±std). "Align." = the dynamics loss trains the shared backbone (representation alignment); "Cond." = the action and video pathways are coupled so the action reads the generated future (explicit conditioning). All means are at 40k steps; std is over 10 trials per task. The **Average** column is the macro-average over the four suites; the best average is in bold.

			SPATIAL		OBJECT		GOAL		LONG		**Average**	
Method	Align.	Cond.	SR	Pr	SR	Pr	SR	Pr	SR	Pr	SR	Pr
Action-only	✗	✗	$0.0_{\pm 0.0}$	$5.0_{\pm 2.5}$	$0.0_{\pm 0.0}$	$11.7_{\pm 2.9}$	$0.0_{\pm 0.0}$	$11.7_{\pm 1.4}$	$0.0_{\pm 0.0}$	$9.2_{\pm 0.7}$	$0.0_{\pm 0.0}$	$9.4_{\pm 1.9}$
Repr. alignment	✓	✗	$3.3_{\pm 5.8}$	$25.0_{\pm 6.6}$	$0.0_{\pm 0.0}$	$7.5_{\pm 2.5}$	$3.3_{\pm 5.8}$	$25.0_{\pm 7.5}$	$0.0_{\pm 0.0}$	$6.2_{\pm 1.9}$	$1.7_{\pm 2.9}$	$15.9_{\pm 4.6}$
Joint denoising	✓	✓	$30.0_{\pm 5.8}$	$75.0_{\pm 2.5}$	$20.0_{\pm 11.5}$	$37.5_{\pm 9.5}$	$40.0_{\pm 15.3}$	$53.3_{\pm 9.8}$	$20.0_{\pm 5.8}$	$57.1_{\pm 2.9}$	$27.5_{\pm 9.6}$	$55.7_{\pm 6.2}$
Self-feedback	✓	✓†	$50.0_{\pm 10.0}$	$77.5_{\pm 5.2}$	$40.0_{\pm 5.8}$	$57.5_{\pm 1.4}$	$40.0_{\pm 10.0}$	$55.8_{\pm 7.7}$	$0.0_{\pm 5.8}$	$51.7_{\pm 2.2}$	$\mathbf{32.5}_{\pm 7.9}$	$\mathbf{60.6}_{\pm 4.1}$

† also feeds the model's own generated frame back as input across control steps (grounded feedback).

metrics. Success rate is the fraction of trials that satisfy the task's goal predicate. Progress is a finer-grained score for partial completion: we split each task's goal into an ordered sequence of stages—for a pick-and-place task, reach → grasp → transport → place—and, from the simulator's ground-truth state, take the furthest stage reached divided by the number of stages. The last stage is the exact success predicate, so progress equals 1 precisely when a trial succeeds, while giving a smoother, lower-variance signal when success is rare (Appendix B gives the full definition). We summarize each method by the macro-average over the four suites; per-suite numbers appear in Table 1, and the generated-future-feedback variant is evaluated with its grounded cross-step feedback enabled.

Training. We train each variant to 40k steps on a single node of eight H100 GPUs: seven run data-parallel training (DeepSpeed ZeRO-2 [14], per-device batch size 1) while the eighth runs validation in parallel, evaluating checkpoints as they are saved. Full hyperparameters are listed in Appendix C.

5 Results

5.1 Quantitative Evaluation

Table 1 reports per-suite and macro-averaged success rate and progress for the four configurations at 40k steps (mean±std over trials). The trend is monotone in the number of active mechanisms and, more importantly, highly non-uniform: the two switches are far from equally important.

The floor: a pure action policy fails. The action-only configuration, which uses neither mechanism, essentially does not solve the tasks (0.0% success; 9.4% progress). With no future-prediction signal shaping the backbone and no forecast to read, the policy rarely advances past the early stages of a task. This sets the floor against which both mechanisms are measured.

Representation alignment alone barely helps. Turning on the dynamics loss while keeping it out of the action path (decoupled) moves success to 1.7% and progress to 15.9%. Relative to the action-only floor this is a small lift—about $+6.5$ progress points and essentially no reliable success—even though the backbone is now trained with the full future-prediction objective. Shaping the representation, on its own, transfers little to control here.

Conditioning supplies almost the entire gain. Coupling the two pathways so the action can read the generated future—with no change to the training objective relative to the decoupled variant—raises success to 27.5% and progress to 55.7%. This single change accounts for roughly $+26$ points of success and $+40$ points of progress, about $15\times$ and $6\times$ what alignment contributed.

Explicit feedback is just as effective. The generated-future-feedback variant, which feeds the model's own finalized forecast back as input across control steps, reaches 32.5% success and 60.6% progress—on par with joint denoising at the same 40k budget. An explicit, inference-time use of the forecast is therefore as effective as reading it within a single pass, corroborating that the operative mechanism is conditioning on the forecast rather than a training-time representation effect.

Training dynamics. Figure 2 plots success rate and progress against training step (10k–40k) for each configuration. The two explicit-conditioning variants (joint and generated-future feedback) pull away from the floor within the first 10–20k steps and stay far ahead on both metrics, while the action-only and decoupled (alignment-only) variants never exceed $\sim$2.5% success or $\sim$16% progress and hug the floor throughout. The two conditioning curves track each other closely; by 40k generated-future feedback is marginally ahead ($32.5\%/60.6\%$ vs. $27.5\%/55.7\%$; Table 1). Joint denoising attains its single best checkpoint slightly earlier, near 36k ($42.5\%/67.7\%$), so the 40k point understates

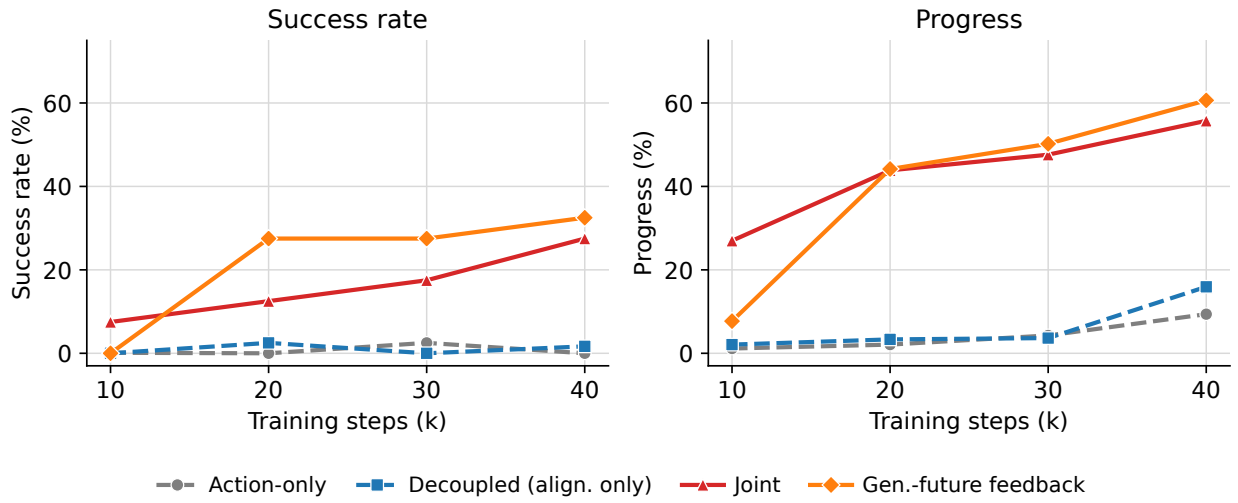


Figure 2: **Success rate and progress over training** (10k–40k), macro-averaged over the four LIBERO suites; one curve per configuration. *Left:* success rate; *right:* progress. The two explicit-conditioning variants—*joint* (red) and *generated-future feedback* (orange), drawn with solid lines—separate from the floor within the first 10–20k steps and remain far above the *action-only* (gray) and *decoupled / alignment-only* (blue) variants, which stay near zero on success and below $\sim$16% progress throughout. The two conditioning curves track each other closely, with generated-future feedback marginally ahead at 40k. Points are at the 10k/20k/30k/40k checkpoints; joint denoising's best checkpoint (near 36k) is not marked, so its 40k point understates its peak. The action-only 10k checkpoint was not separately evaluated and is shown here as a floor-level estimate.

its peak. The separation between the conditioning and non-conditioning groups appears early and is stable across checkpoints.

The table supports a simple additive reading of the two switches. Decomposing the macro-averaged progress relative to the action-only floor (9.4%): turning on alignment adds $+6.5$ points (action-only $\rightarrow$ decoupled), whereas coupling the pathways adds a further $+39.8$ points (decoupled $\rightarrow$ joint). The same pattern holds for success rate, where alignment adds ≈ 1.7 points and conditioning adds ≈ 25.8. Conditioning thus contributes roughly $6\times$ (progress) to $15\times$ (success) what alignment contributes. The explicit-feedback variant slightly exceeds the joint model at this budget (60.6 vs. 55.7 progress; 32.5 vs. 27.5 success) while using the generated future only as an inference-time input, reinforcing that the gain is carried by what the action reads, not by how the backbone was shaped.

Where the policies stall: a grasp bottleneck. Decomposing the progress metric by stage (Appendix A.1) localizes where each policy stalls and what conditioning unlocks. On the four-stage pick-and-place sub-goals that dominate the suites (Table 3), every variant reaches toward the source object at a non-trivial rate, but they separate sharply at the grasp: the action-only policy approaches on 24.8% of sub-goals yet grasps on only 3.1%, and representation alignment raises the grasp rate only to 15.5%. Explicit conditioning breaks this bottleneck—joint denoising and self-feedback grasp on 62.8% and 55.8% of sub-goals, a $20\times$ and $18\times$ increase over the action-only floor—and from there carry an almost identical fraction through transport (48.8% for both) and placement (34.9% for both). The per-episode distribution (Table 2) shows the same wall: the non-conditioning variants almost never pass the halfway mark ($\leq 18.3\%$ of episodes reach $\geq 50\%$ progress, and no action-only episode reaches $\geq 75\%$), whereas both conditioning variants reach $\geq 75\%$ progress on 45% of episodes. Conditioning therefore does not scale progress uniformly; it removes a specific failure mode, turning policies that merely reach for objects into policies that can pick them up.

Alignment's small lift is uneven across suites. The $+6.5$-point average progress gain from representation alignment is not distributed evenly. Relative to the action-only floor it is concentrated on SPATIAL ($+20.0$) and GOAL ($+13.3$) and is in fact slightly negative on OBJECT (-4.2) and LONG (-3.0; Table 1), consistent with a marginal, non-robust effect rather than a reliable transfer to control. Explicit conditioning, by contrast, improves every suite over alignment, with the largest gains on LONG ($+50.9$) and SPATIAL ($+50.0$).

5.2 Qualitative Analysis

Each mechanism reaches one stage further. Figure 3 makes the grasp bottleneck concrete on a single SPATIAL task run from one initial state. Read top to bottom, the furthest stage reached climbs exactly one rung as each mechanism is switched on: the action-only policy hovers above the table and never approaches the bowl; adding representation

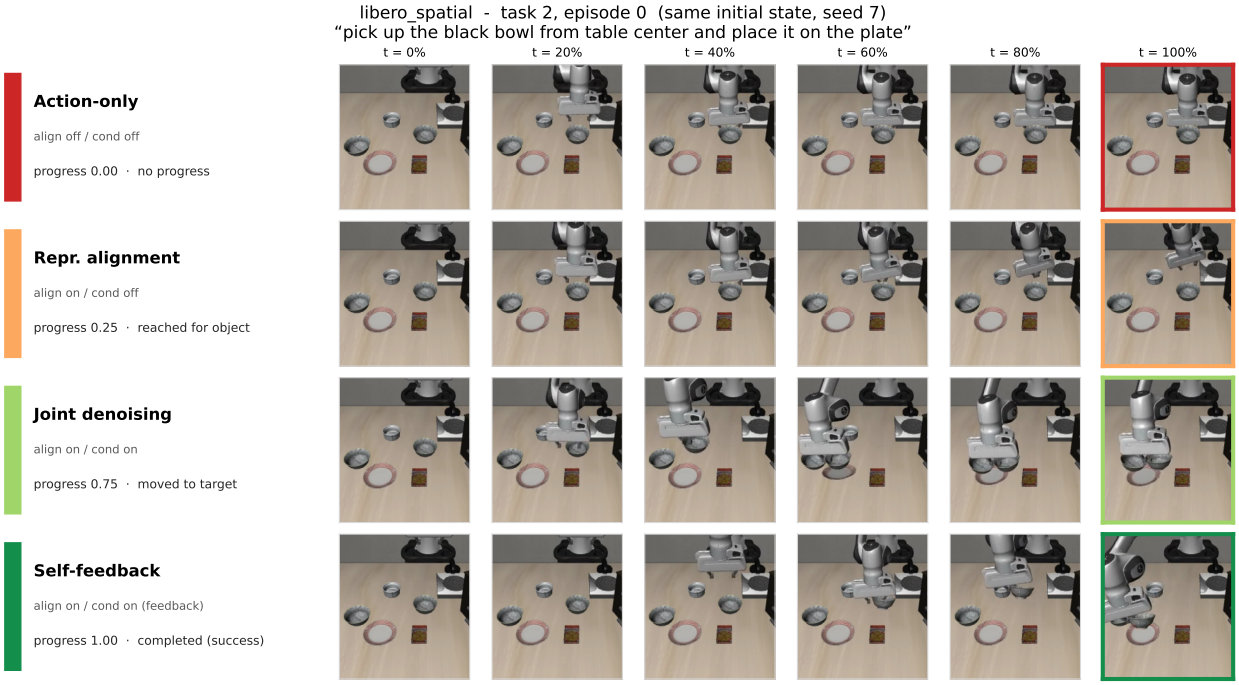
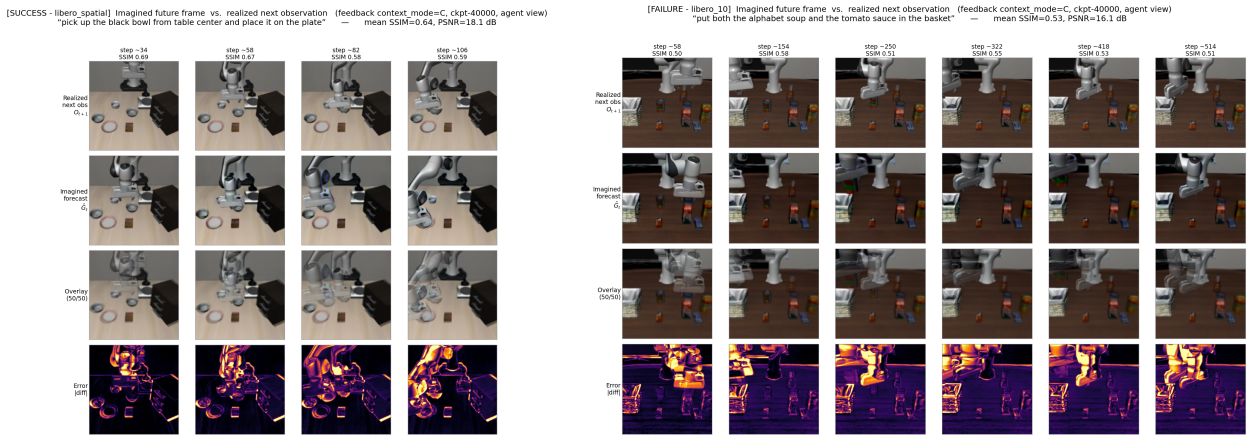


Figure 3: **One scene, four mechanisms.** Rollouts on the same SPATIAL task and initial state (*“pick up the black bowl from table center and place it on the plate”*; episode 0, seed 7), one row per configuration in additive order, with frames sampled evenly over the episode and the final frame outlined. The furthest stage reached (left) climbs monotonically as mechanisms are added: the action-only policy never descends to the object (progress 0.00), representation alignment reaches for it but does not grasp (0.25), and explicit conditioning grasps and transports—joint denoising stalls at placement (0.75) while self-feedback completes the place (1.00). The figure localizes the grasp bottleneck of Section 5: the non-conditioning rows stall at or before the grasp, exactly where the stage statistics (Table 3) drop.

alignment lets it reach the object but not close the gripper; and adding explicit conditioning breaks the grasp barrier, with joint denoising carrying the bowl toward the plate and self-feedback completing the placement. The same scene solved to four different depths mirrors the aggregate stage statistics (Table 3), where the non-conditioning variants collapse at `grasp_src`.

Self-feedback lives or dies by its own forecast. Of the two conditioning variants, self-feedback is the most heavily dependent on the video generation: it does not read a forecast that is regenerated and discarded each step (as joint denoising does) but feeds its own imagined frame back as a context input across control steps, so the fidelity of the generated video directly gates the closed loop. Figure 4 exposes this dependence by overlaying, at each control step, the frame the model imagined one block ahead ($\hat{G}_t$, the frontier it feeds back as conditioning) against the realized next observation it actually receives at the following query (O_{t+1}); columns are successive timesteps and the rows show the realized frame, the imagined frame, their $50/50$ overlay, and a per-pixel error heat-map annotated with SSIM. In the successful SPATIAL rollout (Figure 4a) the forecast stays locked onto reality: imagined and realized frames are nearly indistinguishable (mean SSIM ≈ 0.64, peaking ≈ 0.69), the static scene—table, plate, bowl, cabinet—is reconstructed almost exactly, and the only error is a thin halo on the fast-moving gripper, so the frame fed back is a faithful stand-in and the policy stays on track. The failed LONG rollout (Figure 4b) shows the opposite, and—crucially—shows that the divergence is *not self-correcting*: rather than re-locking onto reality the forecast stays decoupled for the rest of the episode (SSIM hovers around 0.53 instead of recovering), the overlay shows persistent ghosting of the arm, and the error spreads from the gripper onto the manipulated objects as the rollout proceeds. Because the policy keeps conditioning on its own drifting prediction—with no signal in the loop to correct the forecast—the error compounds and never comes back, so the same feedback that supplies the gain when the forecast is good becomes a trap once it is not. This is the failure mode that most directly limits self-feedback (and the one joint denoising sidesteps by re-anchoring on the real observation each step), and it points to forecast fidelity and drift-recovery as the levers for improving the variant that drives the gains in Section 5.



(a) Successful SPATIAL rollout (mean SSIM 0.64). **(b)** Failed LONG rollout (SSIM ≈ 0.53, no recovery).

Figure 4: **Imagined vs. realized forecast for self-feedback.** At each control step the frame the model imagined one block ahead ($\hat{G}_t$) is compared with the realized next observation (O_{t+1}); within each panel the rows are the realized frame, the imagined frame, their $50/50$ overlay, and the per-pixel error $|\Delta|$, with per-column SSIM. **(a)** On a successful SPATIAL rollout (*"pick up the black bowl from table center and place it on the plate"*) the forecast matches reality almost exactly (mean SSIM 0.64); residual error is a thin halo on the moving gripper, so the fed-back frame is a faithful conditioning signal. **(b)** On a failed LONG rollout (*"put both the alphabet soup and the tomato sauce in the basket"*) the forecast drifts and does not recover (SSIM ≈ 0.53 throughout): the arm ghosts in the overlay and the error spreads from the gripper onto the manipulated objects. Because self-feedback conditions on its own prediction, the divergence is not self-correcting.

6 Discussion

Interpretation. Taken together, our experiments give a clear answer to the question we set out to study: in this World-Action Model on LIBERO, generated futures help primarily as an input that the action reads, not as a training-time regularizer on the shared representation. Starting from a pure action policy that essentially fails (0.0% success, 9.4% progress), adding a strong future-prediction objective that back-propagates through the shared backbone—representation alignment—barely lifts the floor (1.7% success, 15.9% progress), even though the backbone is now trained to predict the future. The moment the action is allowed to attend to the generated future, performance jumps by more than an order of magnitude in success rate (joint denoising $27.5\%/55.7\%$, self-feedback $32.5\%/60.6\%$), despite the training objective being unchanged from the alignment-only variant. That the two conditioning variants—one reading a noisy within-pass forecast, the other feeding its own finalized frame back across control steps—land in the same range tells us that the gain is carried by the action reading the forecast at all, rather than by the particular way it is supplied. The stage-decomposed analysis sharpens this: conditioning does not scale progress uniformly but removes a specific failure mode, breaking the grasp bottleneck that traps the non-conditioning variants and turning policies that merely reach for objects into policies that can pick them up. The practical implication is that the generator belongs on the inference critical path. Dropping it to keep only a "cheap" action policy—the move that representation alignment alone would justify—would forfeit nearly all of the gain here.

Future work. Our study isolates the alignment-versus-conditioning mechanism within a single backbone on a single benchmark family, so the natural next step is to test whether the same pattern holds more broadly. A parallel study could repeat the two-switch decomposition on a different class of action model—for example a flow-matching VLA such as π_0/`openpi` [1]—and across additional benchmarks and embodiments. If explicit conditioning consistently dominates representation alignment across architectures and task suites, that would point to a generalizable principle for robot policy learning: supply the generated future to the action pathway as an input rather than relying on it as an auxiliary representation-shaping loss. Running these parallel studies was out of scope here, primarily because of compute—each variant requires full fine-tuning of a 5B-parameter backbone, and replicating the full sweep across several models and benchmarks was beyond the budget available for a course project. The numbers we do report are, moreover, conservative. As Figure 2 shows, none of the variants has saturated at our 40k-step budget—the conditioning curves are still climbing on both metrics—so longer training would likely raise the absolute scores. Training also used a small effective batch size (per-device batch size 1 across seven data-parallel GPUs) relative to the much larger batches typically used when pretraining models of this scale from scratch; a larger batch together with a longer schedule is a straightforward axis along which these results could improve and would make the absolute success rates more competitive.

7 Conclusion

We set out to explain why predicting a generated future helps robot policies, separating two mechanisms—representation alignment and explicit conditioning—that are normally entangled, and isolating them within a single joint video–action model through two minimal switches. Across four LIBERO suites the answer is unambiguous: representation alignment alone barely improves over a collapsed action-only policy, whereas letting the action read the generated future supplies almost the entire gain, in two distinct forms of conditioning that perform comparably. The generated future earns its keep as an input on the inference critical path, not as a training-time scaffold that can be discarded at deployment. For this World-Action Model, the productive direction is therefore not to drop the generator but to make it faster and more accurate and to couple it tightly to the action pathway. Whether the same conclusion generalizes to other action models and benchmarks is a question we believe is worth answering at scale.

8 Team Contributions

- **Minyeong Kim:** Sole author. Formulated the alignment-versus-conditioning question and the two-switch decomposition; implemented the variants; ran all training and LIBERO evaluations; and wrote the report.

Changes from Proposal The project narrowed from a broad "do world models help policies?" question to a controlled mechanism attribution—representation alignment versus explicit conditioning—enabled by the two-switch design. We also added the explicit generated-future-feedback variant and adopted the soft, stage-decomposed progress metric (alongside binary success) to obtain a lower-variance signal at low success rates.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv:2410.24164 [cs.RO]

[2] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. 2023. Zero-Shot Robotic Manipulation with Pretrained Image-Editing Diffusion Models. arXiv:2310.10639 [cs.RO]

[3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2023. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. arXiv:2303.04137 [cs.RO]

[4] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. 2023. Learning Universal Policies via Text-Guided Video Generation. arXiv:2302.00111 [cs.LG]

[5] Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. 2024. Visual Policy Learning via Joint Denoising Process. arXiv:2411.18179 [cs.RO]

[6] Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. 2025. Video Generators are Robot Policies. arXiv:2508.00795 [cs.RO]

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

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

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

[10] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG]

[11] William Peebles and Saining Xie. 2022. Scalable Diffusion Models with Transformers. arXiv:2212.09748 [cs.CV]

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV]

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs.LG]

[14] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. arXiv:1910.02054 [cs.LG]

[15] Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. 2025. VideoVLA: Video Generators Can Be Generalizable Robot Manipulators. arXiv:2512.06963 [cs.RO]

[16] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A Physics Engine for Model-Based Control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Vilamoura-Algarve, Portugal, 5026–5033. doi:10.1109/IROS.2012.6386109

[17] Wan-AI. 2025. Wan2.2-TI2V-5B. https://huggingface.co/Wan-AI/Wan2.2-TI2V-5B.

[18] Wan Team. 2025. Wan: Open and Advanced Large-Scale Video Generative Models. arXiv:2503.20314 [cs.CV]

[19] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. 2023. Unleashing Large-Scale Video Generative Pre-training for Visual Robot Manipulation. arXiv:2312.13139 [cs.RO]

[20] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Jim Fan, and Joel Jang. 2026. World Action Models are Zero-shot Policies. arXiv:2602.15922 [cs.RO]

A Additional Experiments

A.1 Progress reach distribution

Beyond the mean progress reported in Table 1, we summarize the full distribution of how far each policy advances at the matched 40k-step budget. Table 2 gives the percentage of evaluation episodes whose final per-episode progress (Eq. (6)) reaches at least a given fraction of a task's stages, pooled over the four suites (equivalently, macro-averaged, since within a method each suite contributes the same number of episodes). The 100% column is identical to the binary success rate. Because tasks have different stage counts ($K_j \in \{2, 3, 4\}$), a fixed progress threshold does not map to a single stage across tasks; Table 3 therefore complements it with a stage-explicit view restricted to the four-stage pick-and-place family (`on`/`in` onto an object or container; 43 sub-goal instances pooled over the four suites), reporting the episode-weighted percentage of sub-goals reaching at least each ordered stage. Its terminal `done` column equals the pick-and-place sub-goal success rate.

Two caveats. First, the four 40k checkpoints were evaluated with different trial budgets—three trials per task for action-only and representation alignment ($N = 120$ episodes), one per task for the two conditioning variants ($N = 40$)—so the conditioning rows are estimated from fewer episodes and the main paper draws its $\pm$std from the nearest three-trial checkpoint (Table 1). Second, these are the same 40k checkpoints as the main table, so the joint variant's stronger 36k checkpoint is not reflected here.

Read together, the two tables expose the grasp bottleneck discussed in Section 5: the non-conditioning variants reach toward objects but rarely grasp them (the action-only grasp rate is only 3.1%, vs. a 24.8% approach rate), and they almost never cross 50% progress, whereas explicit conditioning lifts grasping by roughly $20\times$ and pushes 45% of episodes past the 75% mark.

Table 2: **Progress reach distribution at 40k steps.** Percentage of evaluation episodes (pooled over the four LIBERO suites) whose final per-episode progress reaches at least the indicated fraction of task stages. N is the number of episodes; the 100% column equals the binary success rate. "Align."/"Cond." as in Table 1.

Method	Align.	Cond.	N	$>0\%$	$\geq 25\%$	$\geq 50\%$	$\geq 75\%$	100%
Action-only	✗	✗	120	34.2	30.8	4.2	0.0	0.0
Repr. alignment	✓	✗	120	36.7	33.3	18.3	7.5	1.7
Joint denoising	✓	✓	40	87.5	85.0	60.0	45.0	27.5
Self-feedback	✓	✓†	40	90.0	90.0	70.0	45.0	32.5

† also feeds the model's own generated frame back as input across control steps.

Table 3: **Pick-and-place stage reach at 40k steps.** For the four-stage pick-and-place sub-goals (43 instances pooled over the four suites), the episode-weighted percentage reaching at least each ordered stage: `approach_src` (reach the source object) → `grasp_src` (grasp it) → `approach_tgt` (transport toward the target) → `done` (place; the exact BDDL predicate). The `done` column equals the pick-and-place sub-goal success rate. The sharp fall at `grasp_src` for the non-conditioning variants is the grasp bottleneck.

Method	Align.	Cond.	`approach_src`	`grasp_src`	`approach_tgt`	`done`
Action-only	✗	✗	24.8	3.1	0.0	0.0
Repr. alignment	✓	✗	29.5	15.5	7.8	1.6
Joint denoising	✓	✓	81.4	62.8	48.8	34.9
Self-feedback	✓	✓†	79.1	55.8	48.8	34.9

† also feeds the model's own generated frame back as input across control steps.

B The Progress Metric

Motivation and scope. Binary success rate is a high-variance signal when success is rare: at our 40k-step budget several variants succeed on few or no trials, so success alone cannot distinguish a policy that wanders aimlessly from one that grasps the right object but fails to place it. The progress score addresses this by awarding ordered partial credit for reaching the intermediate stages of a task. It is computed entirely on the simulation client from the ground-truth MuJoCo state together with LIBERO's own predicate and grasp helpers; the policy server is never consulted, so the metric introduces no model-side assumptions and is applied identically to all variants.

Goal decomposition. Each LIBERO task specifies a BDDL goal that is a conjunction of m sub-goals, $G = \bigwedge_{j=1}^{m} g_j$, where each sub-goal $g_j = p_j(\mathbf{o}_j)$ applies a predicate p_j to one or two object arguments $\mathbf{o}_j$. The entire benchmark uses only six predicates, which we group into three families, each with its own ordered list of stages $\mathcal{S}_j = (s_{j,1}, \ldots, s_{j,K_j})$ (Table 4). Pick-and-place predicates (`on`, `in`) normally expand to four stages, but degenerate to a three-stage push template—dropping the grasp—when the target is a table floor zone (an argument prefixed `main_table_`) or the task language begins with "push", since such objects are shoved rather than lifted. Articulation predicates expand to two stages.

Table 4: **Stage templates by sub-goal family.** The terminal `done` stage of every family is the exact BDDL predicate p_j, so reaching all terminal stages is identical to the environment's own success check.

Family	Predicates	Ordered stages (K_j)
Pick-and-place	`on`, `in` (onto object/container)	`approach_src` → `grasp_src` → `approach_tgt` → `done` (4)
Push	`on`, `in` (into floor zone / "push")	`approach_src` → `near_tgt` → `done` (3)
Articulation	`open`, `close`, `turnon`, `turnoff`	`approach` → `done` (2)

Instantaneous stage. At each executed simulation step t we compute, for every sub-goal, the highest stage index currently consistent with the state, $\sigma_j(t) \in \{0, \ldots, K_j\}$. Let x_{eef} be the end-effector position, x_a, x_b the positions of the source and target arguments (movable-body, region-site, or fixture centres), $\text{grasp}(a)$ LIBERO's threshold-free fingerpad-contact test, and p_j the exact predicate evaluator. With approach threshold $d_{\text{app}} = 0.07$ m and target threshold $d_{\text{tgt}} = 0.12$ m, the pick-and-place rule is

$$\sigma_j(t) = \begin{cases} 4 & p_j \text{ holds}, \\ 3 & \big(\text{grasp}(a) \ \vee \ M_j \geq 2\big) \ \wedge \ \|x_a - x_b\| < d_{\text{tgt}}, \\ 2 & \text{grasp}(a), \\ 1 & \|x_{\text{eef}} - x_a\| < d_{\text{app}}, \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

evaluated top-down (the first satisfied case wins), where M_j is the latched maximum defined below. The push family replaces the grasp test by a second proximity test: $\sigma_j = 3$ when p_j holds, $\sigma_j = 2$ once ($\|x_{\text{eef}} - x_a\| < d_{\text{app}} \ \vee \ M_j \geq 1$) $\wedge \|x_a - x_b\| < d_{\text{tgt}}$, and $\sigma_j = 1$ once $\|x_{\text{eef}} - x_a\| < d_{\text{app}}$. The articulation family has a single geometric stage, $\sigma_j = 1$ when $\|x_{\text{eef}} - x_a\| < d_{\text{tgt}}$, before the exact predicate at $\sigma_j = 2$.

Latching: strict ordering with soft credit. Stages are latched monotonically over the episode,

$$M_j = \max_t \ \sigma_j(t) \in \{0, \ldots, K_j\}, \tag{5}$$

so once a stage is reached it is never lost. Two rules keep the latch robust. Soft credit: whenever the exact predicate p_j becomes true, Eq. (4) returns the terminal index K_j and credits all earlier stages at once—so a task solved without a heuristic ever firing (e.g. a fast grasp that the approach test missed) still scores full progress; likewise a detected grasp implies the object was approached. Strict ordering: the target-approach stage is gated on having already held the source ($\mathrm{grasp}(a) \vee M_j \geq 2$ in Eq. (4)), so source–target proximity at the start of an episode—before the object is ever picked up—cannot be mistaken for progress.

Episode and aggregate progress. Per episode, progress is the mean of the per-sub-goal stage fractions, a parallel aggregation over the conjuncts of the goal,

$$\mathrm{prog}_{\mathrm{ep}} = \frac{1}{m} \sum_{j=1}^{m} \frac{M_j}{K_j} \in [0, 1]. \tag{6}$$

A task's `mean_episode_progress` averages $\mathrm{prog}_{\mathrm{ep}}$ over its N trials. We additionally report, per sub-goal, the reach fraction $r_j[k] = \frac{1}{N} \sum_e \mathbf{1}\left[M_j^{(e)} \geq k\right]$—the fraction of trials reaching at least stage k—whose final entry $r_j[K_j]$ equals that sub-goal's success rate. A suite's `overall_mean_progress` pools $\mathrm{prog}_{\mathrm{ep}}$ over all episodes of all its tasks, and the single number reported in the main paper is the macro-average of `overall_mean_progress` across the four suites (the same way success rate is summarized).

Success equivalence and threshold robustness. Because every family's terminal `done` stage is the exact BDDL predicate, the environment declares success precisely when all conjuncts hold ($\bigwedge_j p_j$), and a rollout halts at the first such success, a perfect score $\mathrm{prog}_{\mathrm{ep}} = 1$ coincides with binary success. The endpoints of the metric are therefore threshold-free: both the zero floor and the unit (success) ceiling are independent of d_{app} and d_{tgt}, which influence only the intermediate partial-credit stages. Moreover the two stages most decisive for credit—`grasp_src` (fingerpad contact) and `done` (the predicate)—use no distance threshold at all. Progress is thus a lower-variance proxy that agrees with binary success at the top while still separating near-misses from total failures—exactly the regime our low-success variants occupy.

Worked examples. On the LONG task "put both the alphabet soup and the tomato sauce in the basket" (two `in` pick-and-place conjuncts, $K_j = 4$ each), a rollout that approaches the first object but never grasps it and never touches the second scores $M = (1, 0)$, i.e. $\mathrm{prog}_{\mathrm{ep}} = \frac{1}{2}(\frac{1}{4} + \frac{0}{4}) = 0.125$. On "put both moka pots on the stove" (two `on` conjuncts plus a `turnon` articulation conjunct, $K = (4, 4, 2)$), a rollout that satisfies only the stove toggle scores $M = (0, 0, 2)$, i.e. $\mathrm{prog}_{\mathrm{ep}} = \frac{1}{3}(0 + 0 + 1) = 0.33$. Both agree with the recorded per-task metrics.

Implementation notes. Progress is updated after every executed action step, following the ten dummy-action settling steps that begin each episode, and including the final step on which the environment reports success (so a success always soft-credits all stages). Scoring is enabled by default (`-progress-scores`); the thresholds are exposed as `-approach-dist` and `-place-dist`, and the metrics—each sub-goal's stage template, its reach fractions, and the per-episode max-stage vectors—are written to `metrics.json` incrementally, so an interrupted run still yields partial results.

C Implementation Details

Backbone and weights. The backbone is Wan2.2-TI2V-5B [17, 18]: a 30-layer diffusion transformer with hidden size 3072 and a 48-channel video VAE, together with a T5 text encoder [13] and the Wan2.1 CLIP image encoder [12]. We full-fine-tune the model; the pretrained encoders bundled with the backbone stay frozen.

Tokens and horizons. Video is 160×320 with two side-by-side views; each 33-frame clip is compressed to 9 latent frames (one context frame and four blocks of two). Per block we use two video frames, one state token, and a 24-action chunk (action horizon 24). Proprioception is 8-dimensional and actions are 7-dimensional, both q01/q99-normalized.

Optimization. Full fine-tuning with AdamW [10] (learning rate 1×10^{-5}, weight decay 1×10^{-5}, warmup ratio 0.05), bf16 precision, and DeepSpeed ZeRO-2 [14] at per-device batch size 1 across seven data-parallel H100 GPUs (seed 42). Each variant is trained to 40k steps with a checkpoint saved every 1000 steps. The self-feedback variant additionally enables the one-block future-frame shift during training and grounded feedback at evaluation, replanning one block per query.

Evaluation harness. Evaluation is client–server: a GPU policy server returns action chunks to a CPU LIBERO/MuJoCo client that scores success and progress. The four suites use their default per-episode step caps, and the action head's causal cache is reset at each episode boundary.