PLAYWAM: LEARNING DYNAMICS AND VALUE IN WORLD ACTION MODELS THROUGH CONTROLLED COUNTERFACTUAL PLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) serve as both dynamics models and action generators. However, this dual role creates conflicting data requirements: dynamics learning benefits from diverse experiences beyond successful execution, whereas action generation requires supervision selected for task success. Our key insight is that simulation enables controlled alternative futures from shared task states, revealing both physical consequences and their task utility. Motivated by this, we introduce **PlayWAM**, which combines branch-specific learning with *controlled counterfactual play*—a simulation-native pipeline for generating comparable alternative futures from shared states. We use state cloning and parallel rollouts with controlled interventions to capture nominal, perturbed, failing, and recovering behaviors. PlayWAM learns dynamics from all recorded branches but imitates only task-successful segments; other observed actions condition future prediction rather than serve as imitation targets. A lightweight value head ranks same-state branches and scores parallel future–action candidates, enabling single-round selection without outer-loop refinement. We evaluate on PlayBench, RoboTwin-Play, LIBERO, and eight real-world tasks. Project page: `https://stellar-wisp-d1ecc3.netlify.app/`

1 INTRODUCTION

World Action Models have recently seen rapid progress in robot learning. They jointly predict future observations and robot actions conditioned on the current context and task instructions, as in DreamTacVLA (Ye et al., 2025), DreamZero (Ye et al., 2026), and FastWAM (Yuan et al., 2026). These two roles introduce a fundamental data tension. Future prediction calls for broad coverage of action-conditioned dynamics (Wu et al., 2025; Lu et al., 2026c; Yang et al., 2024; Zhu et al., 2025b), including counterfactual variations, external disturbances, and exploratory interactions. Policy learning (Li et al., 2026a; Shen et al., 2025; Ning et al., 2023; Chi et al., 2023), in contrast, relies on successful demonstrations that explicitly supervise actions aligned with the task instruction. We therefore ask: **How can a WAM learn future dynamics from diverse physical experiences while preserving its ability to generate actions that successfully accomplish the instructed task?**

Achieving this balance is challenging under current data collection and inference practices. Existing robot demonstration datasets (Lu et al., 2024; 2026b; O'Neill et al., 2024; Khazatsky et al., 2024; Walke et al., 2023) mostly contain successful executions, leaving counterfactual variations, external disturbances, and broader action-conditioned dynamics underrepresented. Meanwhile, MPC-style planning often uses separate reward or value models for costly outer-loop optimization (Wu et al., 2025; Li et al., 2026b), adding training overhead and inference latency. We therefore propose **PlayWAM**, combining *controlled counterfactual play* with *parallel branch generation and value-based selection*. Rather than collect isolated demonstrations, our play mechanism constructs paired action–future branches from shared task states through controlled interventions, making continuations directly comparable. This structured experience provides complementary supervision: individual trajectories reveal action-conditioned dynamics, while comparisons across branches reveal which continuations support task success. With this supervision, PlayWAM jointly generates multiple actions and their anticipated futures in parallel, then uses a lightweight latent value head to select among them without outer-loop iterative refinement. During training, all play branches super-

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

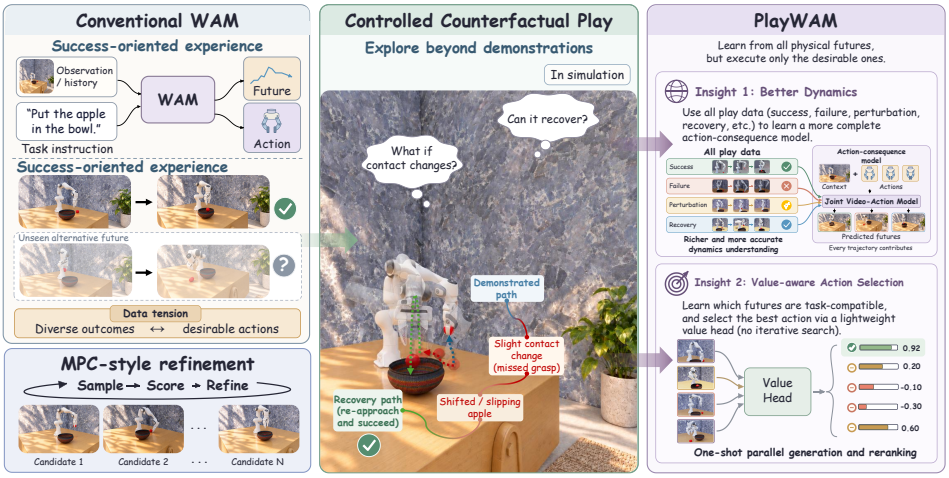


Figure 1: **Overview.** *Left:* successful-only training shows one favorable continuation per state. *Middle:* controlled counterfactual play in simulation branches a shared state into nominal, CF, DS, NC, and recovery futures. *Right:* PlayWAM learns dynamics from all branches, imitates only successful segments, and selects among parallel candidates with a lightweight value head.

vise action-conditioned dynamics, only task-successful segments supervise action generation, and same-state branch comparisons teach task-conditioned value preferences. At inference, PlayWAM scores completed branches once and executes the highest-scoring candidate's action prefix. It thus learns *what can happen under candidate actions* and *which futures are worth pursuing*.

Our simulation-native pipeline generates this supervision safely at scale through state cloning, asynchronous parallel rollouts, automatic outcome annotation, and branch-group streaming. *Nominal* and *Recovery* provide successful references under normal and off-nominal conditions. *Counterfactual (CF)* interventions alter task-relevant configurations, whereas *Disturbance (DS)* introduces transient execution-time perturbations. Either may succeed or fail. *No-Correction (NC)* records a normal failure without retry, whereas *Recovery* records a successful retry after failure; their matched anchor state enables a controlled retry-versus-no-retry comparison. This separates *how deviations arise* from *how they are resolved*, supporting broad dynamics learning and task-directed selection.

To evaluate both task competence and robustness, we introduce a play-based benchmark comprising eight tabletop and four mobile manipulation tasks and extend RoboTwin (Chen et al., 2025) with the same controlled play mechanism. Both provide paired play data and evaluation protocols for counterfactual state changes and execution-time disturbances, enabling systematic assessment of robustness and recovery beyond nominal execution. On our benchmark, PlayWAM achieves higher task success under these interventions than strong WAM baselines while preserving nominal performance and remaining substantially faster than iterative value-guided planning. On RoboTwin, training PlayWAM with controlled play improves robustness over nominal-only training, demonstrating the value of our data mechanism beyond our simulator. Ablations on LIBERO (Liu et al., 2023) further show that value-aware selection improves task success with training demonstrations held fixed, isolating the method's contribution from additional play data. On eight real-world bimanual tasks, simulated PlayData further improves physical execution when combined with nominal real-world demonstrations. Together, these evaluations support our data mechanism and model design across tabletop and mobile manipulation with single-arm and bimanual robots.

Our contributions are summarized as follows:

- We introduce **controlled counterfactual play**, generating paired futures from shared simulation states through structured interventions and disturbances. We instantiate it with over 1,000 hours of automatically annotated interaction for 32 TableTop and Mobile tasks; trajectories reveal action-conditioned dynamics, while comparisons provide task-conditioned value supervision.
- We propose **PlayWAM** for branch-specific learning and single-round action selection. PlayWAM learns dynamics from all branches while imitating only successful segments, and ranks parallel candidates with a lightweight value head, avoiding outer-loop iterative optimization.

- **We introduce a play-based robustness benchmark and extend RoboTwin with controlled play and disturbance evaluation.** PlayWAM improves robustness on both benchmarks and task success on LIBERO, remains substantially faster than iterative value-guided planning, and transfers simulation-generated PlayData to eight real-world bimanual tasks.

2 Related Work

World Models and Test-Time Guidance. Unified video–action models couple future prediction with policy learning (Zhu et al., 2025a; Ye et al., 2025; 2026; Yuan et al., 2026). Model-based planning and value or verifier guidance rank or refine candidate actions using predicted futures (Qi et al., 2026; Nakamoto et al., 2024; Kwok et al., 2025; Jain et al., 2025; Li et al., 2026b; Chua et al., 2018; Janner et al., 2022). They commonly add repeated proposal refinement, verifier calls, or iterative latent optimization at deployment. PlayWAM generates candidates in one batched round and scores their final latents once, avoiding outer-loop refinement.

Play, Corrective Data, and Learning from Failure. Play and synthetic demonstration systems expand behavioral coverage (Yin et al., 2026; Wang et al., 2023; Lynch et al., 2020; Mandlekar et al., 2023). DAgger, DART, IntervenGen, and CCIL collect or synthesize corrective experience (Ross et al., 2011; Laskey et al., 2017; Hoque et al., 2024; Ke et al., 2024), while AWR, IQL, and RECAP use value or advantage weighting to learn from off-policy data (Peng et al., 2019; Kostrikov et al., 2022; Physical Intelligence et al., 2025). Failure-aware imitation further ranks degraded demonstrations or learns task progress from unsuccessful rollouts (Brown et al., 2020; Huang et al., 2025). Unlike unstructured teleoperated play, PlayWAM uses task-conditioned scripts to construct same-anchor alternatives and routes them to dynamics, action, and value supervision by outcome.

3 Problem Definition

Play mechanism and branch-structured data. Our play mechanism starts from a task-relevant physical state s_i, clones the simulator state, and applies controlled interventions or alternative continuation strategies to explore different possible futures. This process produces nominal, failure, disturbance, and recovery experiences that are not treated as independent trajectories, but are organized into a branch group

$$\mathcal{G}_i = \left(c_i, s_i, \left\{ \delta_i^{(b)}, a_i^{(b)}, \tau_i^{(b)}, Y_i^{(b)} \right\}_{b=1}^{B_i} \right), \tag{1}$$

where c_i denotes the shared task context, $\delta_i^{(b)}$ is the controlled intervention, $a_i^{(b)}$ is the subsequent action sequence, $\tau_i^{(b)}$ is the resulting future trajectory, and $Y_i^{(b)}$ denotes its binary task outcome. We call the shared physical state and task context the *anchor*. A *branch* is one trajectory continuation from this anchor, and a *branch group* is the set of branches generated from the same anchor. Their differing continuations provide comparable alternative futures for learning.

World–action modeling and value-based selection. A world–action model (WAM) jointly generates future observations and actions. Given $c_t = (o_{t-L:t}, q_t, g)$, comprising visual history, proprioception, and a language goal, our shared WAM F_θ generates B candidates over horizon H:

$$\hat{\xi}_t^b = \left(\hat{o}_{t+1:t+H}^b,\, \hat{a}_{t:t+H-1}^b \right), \qquad b = 1, \ldots, B. \tag{2}$$

From play groups $\mathcal{G}_i$, we learn action-conditioned dynamics from all recorded branches while imitating only desirable action segments. Physical plausibility alone, however, does not imply task success. We therefore learn within-group outcome preferences with a value head V_ϕ, which reads the pooled contextualized future–action representation h_t^b rather than generating separate value latents:

$$\hat{v}_t^b = V_\phi\big(\mathrm{sg}(h_t^b)\big), \qquad b^* = \arg\max_{b \in \{1,\ldots,B\}} \hat{v}_t^b. \tag{3}$$

Because the WAM hidden states are conditioned on c_t, h_t^b already encodes the visual history, robot state, and task goal. Here, sg blocks value gradients to the WAM. We execute the selected prefix $a_t^{\mathrm{exec}} = \hat{a}_{t:t+h_{\mathrm{exec}}-1}^{b^*}$, with $h_{\mathrm{exec}} \le H$, and update the context with real observations for closed-loop control. All candidates are fully generated in one parallel round and scored once in latent space, without decoding candidate videos or performing iterative latent-distribution refinement.

4 CONTROLLED COUNTERFACTUAL PLAY

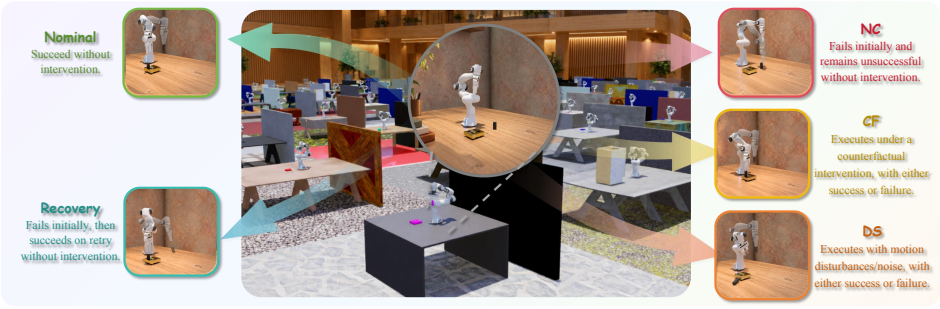


Figure 2: **Play mechanism.** Parallel collection produces branch groups with Nominal, Recovery, No-Correction (NC), Counterfactual Intervention (CF), and Disturbance (DS) trajectories.

A single execution reveals one outcome but not its alternatives. We therefore introduce controlled counterfactual play, a five-mode pipeline that clones a task-relevant state into matched continuations, providing physical evidence within branches and task-value comparisons across them (Fig. 2). Here, CF denotes the task-level intervention mode and DS the execution-time disturbance mode.

4.1 CONTROLLED PLAY MECHANISM

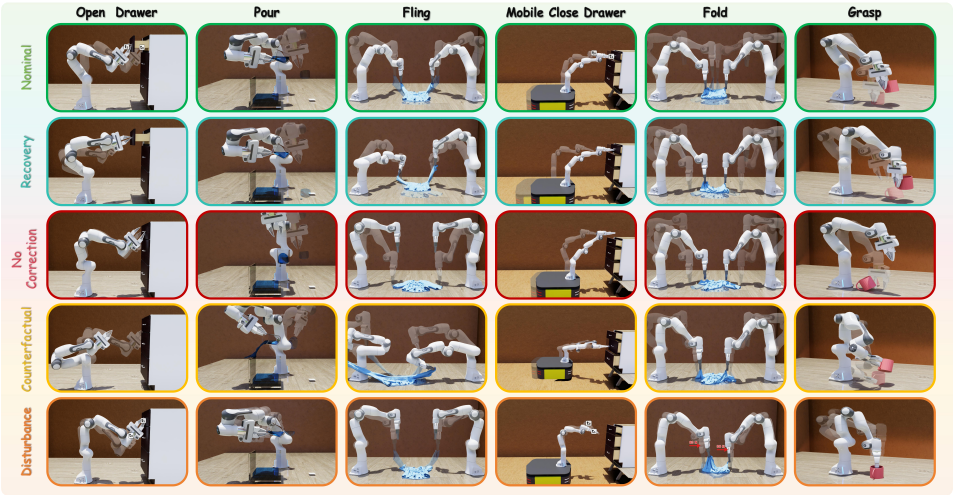


Figure 3: **Examples of the five PlayData modes.** Six same-anchor branch groups.

Five matched continuations. From a shared prefix, we snapshot a task-relevant anchor state and launch five continuation modes from the same physical state and task context. *Nominal* records standard success; *Recovery* succeeds after an initial failed attempt, whereas *NC* preserves the failure without retry. *CF* changes task-relevant configurations and *DS* introduces execution-time disturbances; either may succeed or fail. Figure 3 shows representative same-anchor groups.

Complementary supervision. All recorded branches supervise dynamics, only task-successful segments supervise action generation, and same-anchor outcome differences supervise value learning. This separates physical evidence from behavioral guidance.

4.2 SCALABLE SIMULATION PIPELINE

Safe and reproducible branching. Simulation enables controlled exploration of failures and disturbances without hardware risk or manual scene resets. For each branch group, we snapshot the anchor state after a shared prefix and restore it independently for every continuation. Their physical state and task context are identical at divergence; mode-specific protocols, controller sampling, and physical interaction yield alternative outcomes.

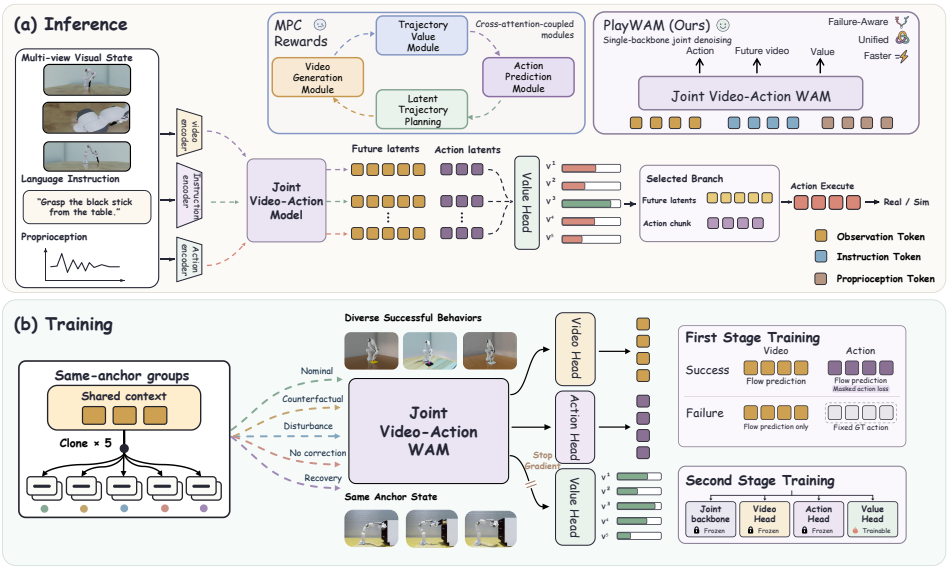


Figure 4: **PlayWAM.** *Top:* B future–action branches are generated in parallel, scored by the value head, and the best action prefix is executed. *Bottom:* all branches supervise future prediction, successful segments supervise actions, and same-state comparisons train the detached value head.

Parallel data collection. We collect branch groups through asynchronous parallel rollouts across TableTop and Mobile tasks. Visual and physical *domain randomization (DR)* (Tobin et al., 2017; Peng et al., 2018) varies conditions across groups while non-intervention factors remain fixed within each group. The pipeline automatically annotates outcomes and discards simulator errors or incomplete recordings. Across 25 TableTop and 7 Mobile tasks, we collect 1,000 five-trajectory branch groups per task—32,000 groups, 160,000 trajectories, and over 1,000 hours of annotated PlayData; Appendix B.1.4 provides the full inventory.

5 PLAYWAM

PlayWAM jointly denoises parallel future–action candidates with a shared WAM. All recorded branches train dynamics, while only desirable actions are imitated. A lightweight value head learns within-group preferences from detached WAM features for task-directed ranking in one pass, avoiding outer-loop latent-distribution optimization. We initialize the video backbone from Genie-Envisioner's pretrained GE-Base-fast checkpoint (Liao et al., 2025), attach a newly initialized action expert and zero-initialized action-to-video adapters, then train in two stages: branch-specific WAM adaptation followed by detached value-head training with the generator frozen.

5.1 JOINT BRANCH GENERATION AND SUPERVISION

Parallel joint generation. We generate multiple candidate branches, pairing alternative action sequences with their corresponding predicted physical consequences. We encode the current context $c_t = (o_{t-L:t}, q_t, g)$ once as C_t, then independently sample B joint noise initializations:

$$\epsilon^b = [\epsilon_z^b, \epsilon_a^b] \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I), \qquad \mathbf{X}_1 = \mathrm{Stack}_{b=1}^{B}(\epsilon^b), \tag{4}$$

where z^b and α^b denote future-video and action latents. At each flow time τ, the shared WAM F_θ processes $\mathbf{X}_\tau = \mathrm{Stack}_{b=1}^{B}([z_\tau^b, \alpha_\tau^b])$ in parallel, predicting both modalities' velocities and branch-specific hidden features:

$$\left\{(\hat{u}_z^b, \hat{u}_a^b, \mathbf{H}_\tau^b)\right\}_{b=1}^{B} = F_\theta(\mathbf{X}_\tau, C_t, \tau). \tag{5}$$

Video and action tokens jointly denoise through within-branch interaction, while branches share parameters and context but retain separate latent states. All branches denoise from $\tau = 1$ to 0 in S steps, yielding $\{(\hat{z}^b, \hat{a}^b, \mathbf{H}^b)\}_{b=1}^{B}$ in one batched round for value-based selection.

Role-dependent world–action supervision. To broaden the WAM's physical understanding without teaching it to reproduce failures, we distinguish physical evidence from behavioral demonstrations. Within each recorded branch group, desirable segments—nominal successes, successful recoveries, and effective CF/DS corrections—form $\mathcal{E}_j$ and supervise both dynamics and action generation. The remaining recorded segments form $\mathcal{D}_j$ and supervise dynamics only, including failed continuations and unsuccessful CF/DS outcomes. This assignment is segment-wise: a recovery's failed attempt remains dynamics-only, while its corrective actions receive joint supervision. For clean latents (z, α) and noise $(\epsilon_z, \epsilon_a) \sim \mathcal{N}(0, I)$, we construct $z_\tau = (1-\tau)z + \tau\epsilon_z$ and $\alpha_\tau = (1-\tau)\alpha + \tau\epsilon_a$ at flow time $\tau \in [0, 1]$. Let $m_{j,b} = \mathbb{1}[b \in \mathcal{E}_j]$ denote action eligibility. Conditioned on the encoded shared context C_j, the shared WAM predicts

$$(\hat{u}_z^{j,b}, \hat{u}_a^{j,b}, \mathbf{H}^{j,b}) = F_\theta\big(z_\tau^{j,b}, m_{j,b}\alpha_\tau^{j,b} + (1 - m_{j,b})\,\mathrm{sg}(\alpha^{j,b}), C_j, \tau\big)\,. \tag{6}$$

Here, sg denotes stop-gradient. Executable segments jointly denoise video and action, whereas dynamics-only segments retain clean, detached actions as conditions for predicting their consequences, not as generation targets. With flow targets (Lipman et al., 2023) $u_z = \epsilon_z - z$ and $u_a = \epsilon_a - \alpha$, let $\ell_z^{j,b} = \|\hat{u}_z^{j,b} - u_z^{j,b}\|_2^2$ and $\ell_a^{j,b} = \|\hat{u}_a^{j,b} - u_a^{j,b}\|_2^2$. We optimize

$$\mathcal{L}_{\mathrm{WAM}} = \mathbb{E}_{j,\tau,\mathrm{noise}}\left[\frac{1}{N_{\mathcal{E}}^j}\sum_{b\in\mathcal{E}_j}\big(\ell_z^{j,b} + \lambda_a \ell_a^{j,b}\big) + \frac{\lambda_{\mathrm{dyn}}}{N_{\mathcal{D}}^j}\sum_{b\in\mathcal{D}_j}\ell_z^{j,b}\right], \tag{7}$$

where $N_{\mathcal{E}}^j = \max(1, |\mathcal{E}_j|)$ and $N_{\mathcal{D}}^j = \max(1, |\mathcal{D}_j|)$ normalize the two roles separately. Dynamics-only updates freeze action-specific modules and omit action loss while updating the shared backbone and world path; we interleave them with executable joint updates to support behavioral preservation.

5.2 Lightweight Branch-Value Modeling

Latent value selection. To distinguish task-effective actions from inaccurate or poorly planned alternatives, we evaluate each candidate through its predicted physical consequences. A lightweight value head reads each completed branch's contextualized video and action features:

$$h_\tau^b = P_z\,\mathrm{sg}(\mathrm{Pool}(\mathbf{H}_{\mathrm{video}}^b)) + P_a\,\mathrm{sg}(\mathrm{Pool}(\mathbf{H}_{\mathrm{action}}^b)) + E_\tau(\tau), \qquad \hat{v}^b = \mathrm{MLP}_\phi(h_\tau^b). \tag{8}$$

The hidden features already encode C_t; at selection, completed branches use $\tau = 0$. The resulting task-conditioned score favors task progress and successful recovery over failed contact, collision, or ineffective continuation. Thus, selection depends on whether an action's anticipated consequences are desirable, rather than merely physically plausible. All candidates are scored in one batched readout without decoding future RGB or sampling separate value latents. We execute the highest-scoring candidate's action prefix and discard the rest:

$$b^* = \arg\max_{b\in\{1,\ldots,B\}} \hat{v}^b, \qquad a_t^{\mathrm{exec}} = \hat{a}_{t:t+h_{\mathrm{exec}}-1}^{b^*}, \qquad h_{\mathrm{exec}} \le H. \tag{9}$$

This ranking enables task-directed selection without outer-loop optimization; after execution, real observations update the context for closed-loop replanning.

Detached branch-value learning. To enable single-round selection without repeated sample–score–refine cycles, we learn task-conditioned branch preferences from controlled play. After WAM adaptation, we freeze F_θ and train V_ϕ on detached branch representations. Within each branch group $\mathcal{G}_j$, simulator utilities $G_{j,b}$ define preference pairs $\mathcal{R}_j = \{(b^+, b^-) \mid G_{j,b^+} > G_{j,b^-} + \epsilon\}$, among utility-labeled branches. We minimize the group-normalized ranking loss:

$$\mathcal{L}_{\mathrm{rank}}^j = \frac{1}{N_{\mathcal{R}}^j}\sum_{(b^+,b^-)\in\mathcal{R}_j}\log\left[1 + \exp\left(-\frac{\hat{v}_{j,b^+} - \hat{v}_{j,b^-}}{\tau_v}\right)\right], \tag{10}$$

where $N_{\mathcal{R}}^j = \max(1, |\mathcal{R}_j|)$ and τ_v is the ranking temperature. These comparisons favor successful recovery over failed continuation from the same disturbed state, rather than treating all off-nominal branches as undesirable. We additionally use a Huber calibration loss $\mathcal{L}_{\mathrm{cal}}$ to align scores with normalized utilities:

$$\mathcal{L}_V = \mathbb{E}_j[\mathcal{L}_{\mathrm{rank}}^j] + \lambda_{\mathrm{cal}}\mathcal{L}_{\mathrm{cal}}, \qquad \nabla_\theta \mathcal{L}_V = 0. \tag{11}$$

Only the value head receives these gradients, leaving the WAM's generative parameters unchanged. The learned readout directly ranks completed candidates at inference, without feeding scores back into repeated resampling or outer-loop latent-distribution optimization.

6 EXPERIMENTS AND BENCHMARKS

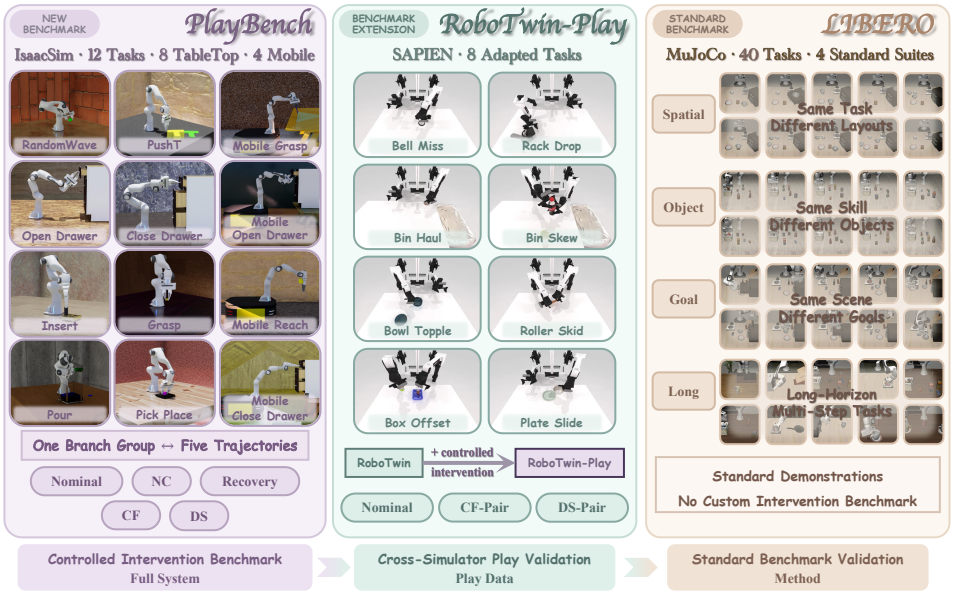


Figure 5: **Three benchmarks, three questions.** PlayBench evaluates the full system (RQ1); RoboTwin-Play isolates structured play data within fixed model families (RQ2); LIBERO isolates value-aware candidate selection on standard data (RQ3).

Our central hypothesis is that branch-specific supervision allows a WAM to learn broader action-conditioned dynamics from failure-containing play while preserving task-directed action generation. Across three benchmarks (Figure 5), we ask:

RQ1: Does failure-containing PlayData improve PlayWAM's success and intervention robustness?
RQ2: Does branch-specific learning from matched failure–recovery play transfer across simulators?
RQ3: Does value-aware candidate selection improve goal-directed control under fixed training data?

6.1 BENCHMARKS AND SETUP

PlayBench (RQ1: full-system evaluation). PlayBench contains eight tabletop and four mobile-manipulation tasks in Isaac Sim. We compare Nominal, which contains 1,000 successful Nominal trajectories per task, with Play-Full, which contains all five trajectories from each of 200 complete branch groups and therefore also totals 1,000 trajectories per task. Play-Success is the naturally occurring successful subset of Play-Full. Evaluation covers standard execution (Nominal), execution after a task-level counterfactual (Post-CF), and execution after a transient disturbance (Post-DS). Appendices E.1 and E.3 provide the task, intervention, and data details.

RoboTwin-Play (RQ2: play-data evaluation). RoboTwin-Play adapts eight bimanual RoboTwin 2.0 tasks (Mu et al., 2025; Chen et al., 2025). Within each model family, Normal uses 150 nominal trajectories per task, while Play-Full uses 50 Nominal, 50 Recovery, and 50 failure trajectories. We keep every pair intact and therefore do not construct a separate Play-Success condition. Architecture, initialization, and evaluation are otherwise fixed. Appendix E.3 provides the full construction and protocol.

LIBERO (RQ3: method evaluation). On the four standard LIBERO suites (Liu et al., 2023), all methods use the same official demonstrations and evaluation protocol without PlayData. Following WAV (Li et al., 2026b), the value head uses only calibration with normalized dense targets derived from discounted rule-based task rewards.

Baselines and variants. We compare with WAV (Li et al., 2026b) and DreamZero-5B (Ye et al., 2026), using Wan2.2-TI2V-5B as its backbone. On PlayBench, DreamZero-5B, WAV, and Play-WAM each use Nominal, Play-Success, and Play-Full; methods receive identical trajectories per

condition while retaining their objectives. For LIBERO, **PlayWAM w/o Value** removes value-based candidate selection and executes one generated continuation. Value-enabled PlayWAM generates B parallel candidates and ranks them with the value head. All variants share the same pretrained initialization. WAV's LIBERO success rates come from its paper.

Metrics and evaluation protocol. We report Nominal, Post-CF, and Post-DS success rates; Robust Avg. averages Post-CF and Post-DS, while LIBERO uses its standard four-suite average. Inference Hz is the reciprocal median wall-clock latency for one 36-step action chunk, measured and compared within each benchmark's native configuration. PlayWAM defaults to $B = 4$ and reports $B \in \{4, 8\}$ on LIBERO; Appendix E.2 gives per-task 100-episode protocols.

6.2 RQ1: PlayWAM Benefits from Failure-Containing PlayData

Table 1: **RQ1: model–data interaction on PlayBench.** Success rate (%) under nominal execution and after CF/DS interventions, Robust Avg. over all 12 tasks, and action-chunk inference rate.

Method	Data	TableTop (8 tasks)			Mobile (4 tasks)			Robust	Efficiency
		Nom. ↑	Post-CF ↑	Post-DS ↑	Nom. ↑	Post-CF ↑	Post-DS ↑	**Avg.** ↑	Hz ↑
DreamZero-5B	Nominal	51.4	26.6	17.6	70.0	34.5	22.3	24.2	**15.23**
	Play-Success	55.0	28.4	20.0	74.3	33.0	23.8	25.6	
	Play-Full	34.3	10.5	4.6	49.5	15.0	6.8	8.7	
WAV	Nominal	56.6	29.5	19.9	71.0	36.5	24.0	26.5	0.58
	Play-Success	59.5	31.5	22.6	79.3	43.0	30.8	30.3	
	Play-Full	31.5	10.3	5.1	53.0	19.8	10.5	10.2	
PlayWAM	Nominal	58.3	30.5	23.8	71.3	39.5	29.8	29.6	6.79
	Play-Success	64.0	38.3	29.1	75.5	46.5	37.8	36.5	
	Play-Full	**65.0**	**52.1**	**44.5**	**80.8**	**69.3**	**59.8**	**53.7**	

PlayBench reveals a model–data interaction. DreamZero-5B and WAV improve from Nominal to Play-Success but degrade on Play-Full, whereas PlayWAM improves monotonically by using failures for dynamics and value learning without imitating their actions. PlayWAM reaches a $\mathbf{53.7}\%$ Robust Avg., versus $\mathbf{30.3}\%$ for the strongest baseline, WAV–Play-Success. Under matched Play-Full data, WAV and DreamZero-5B fall to $\mathbf{10.2}\%$ and $\mathbf{8.7}\%$, respectively. PlayWAM runs at 6.79 Hz—$11.7\times$ faster than WAV's iterative value-guided inference.

6.3 RQ2: Paired Play Transfers across Simulators

Table 2: **RQ2: paired play on RoboTwin-Play.** Eight-task success rate (%)

Method	Data	Nom. ↑	Post-CF ↑	Post-DS ↑	**Robust Avg.** ↑	Hz ↑
DreamZero-5B	Normal	71.4	28.1	25.3	26.7	**24.66**
	Play-Full	60.4	13.4	10.6	12.0	
WAV	Normal	74.5	32.4	26.4	29.4	0.98
	Play-Full	59.5	16.1	9.8	12.9	
PlayWAM	Normal	78.1	38.0	31.1	34.6	11.75
	Play-Full	**80.1**	**66.0**	**63.8**	**64.9**	

RQ2 tests whether the model–data interaction transfers to an independent bimanual simulator while holding each method's trajectory count fixed. Replacing 150 Normal trajectories with 50 Nominal, 50 Recovery, and 50 failure trajectories raises PlayWAM's Robust Avg. from 34.6% to $\mathbf{64.9}\%$ ($+\mathbf{30.3}$ points), while lowering WAV from 29.4% to $\mathbf{12.9}\%$ ($-\mathbf{16.5}$ points). Thus, complete paired-play data improves PlayWAM beyond merely adding examples; we do not attribute the entire gain to failed trajectories alone.

6.4 RQ3: Value Selection Recovers Most Oracle Headroom

RQ3 fixes training data to standard LIBERO demonstrations, isolating value-aware selection from PlayData. Direct execution obtains 98.25%; value selection reaches $\mathbf{98.9}\%$ at $B = 4$ and $\mathbf{99.0}\%$ at $B = 8$, whose lower 2.35 Hz rate favors $B = 4$. As LIBERO is near ceiling, we further analyze identical PlayBench candidate sets: learned value raises Robust Avg. from 46.7% under random

selection to **53.7**%, near the 54.9% simulator-oracle upper bound, recovering 85% of the available selection headroom (Appendix E.4.2).

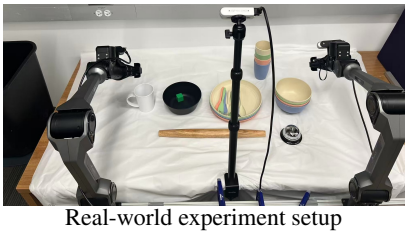
Table 3: **RQ3: value-aware candidate selection on LIBERO.** Fixed data; success rate (%).

Method	Selection	B	Spatial ↑	Object ↑	Goal ↑	Long ↑	**Avg.** ↑	Hz ↑
DreamZero-5B	Direct	–	97.8	98.2	96.2	94.6	96.7	12.69
WAV	Iterative value	–	99.6	**100.0**	98.6	94.4	98.1	0.33
PlayWAM w/o Value	Direct	–	99.0	99.8	97.4	96.8	98.25	**16.10**
PlayWAM	**Value**	**4**	99.6	99.8	**99.2**	97.0	98.9	4.03
PlayWAM	Value	8	99.6	99.8	**99.2**	**97.4**	**99.0**	2.35

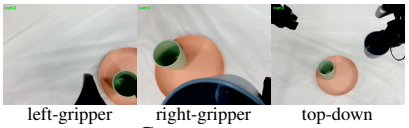
Ablations. On PlayBench, PlayWAM reaches a 53.7% Robust Avg. All-action imitation drops to 22.4%, confirming that failure actions should remain dynamics and value evidence rather than imitation targets. Shuffling branch groups reduces it to 47.7%, supporting same-anchor comparisons; replacing Recovery with count-matched Nominal data gives 49.9%, showing that corrective experience adds information beyond data volume. The 31.3-point loss identifies supervision routing as the dominant factor, with pairing and recovery remaining complementary. Appendix E.4 reports value-selection and intervention analyses.

6.5 Real-World Experiments

Setup. We test eight RoboTwin-Play tasks on a dual-leader/dual-follower Piper-based ALOHA; Appendix E.1.5 shows all task instances. Each model uses 100 successful nominal real-world demonstrations and 200 simulated trajectories per task; the simulated set is success-only for DreamZero and PlayWAM Success-only, and structured PlayData for PlayWAM Full.



Real-world experiment setup



left-gripper right-gripper top-down

Camera setup

Task	DreamZero	PlayWAM Success-only	PlayWAM Full
Click Bell	10/20	14/20	**17/20**
Place Container Plate	12/20	15/20	**17/20**
Hanging Mug	5/20	10/20	**14/20**
Dump Bin Bigbin	8/20	12/20	**15/20**
Put Bottles Dustbin	11/20	13/20	**15/20**
Stack Bowls Three	7/20	11/20	**14/20**
Grab Roller	12/20	15/20	**19/20**
Place Cans Plasticbox	9/20	12/20	**16/20**
Total	74/160	102/160	**127/160**

Figure 6: **Real-world evaluation.** *Left:* our Piper-based ALOHA platform. *Right:* successful executions out of 20 trials per task. DreamZero and PlayWAM Success-only use the same nominal real-world and successful simulated trajectories; PlayWAM Full replaces the simulated subset with structured PlayData while keeping the real-world demonstrations fixed.

Results. Under identical nominal real-world and successful simulated data, PlayWAM Success-only outperforms DreamZero (Fig. 6), providing a same-data comparison. Within PlayWAM, Full improves over Success-only with real-world demonstrations fixed, isolating the effect of simulated-data composition. Structured CF/DS, failure, and recovery experience therefore transfers to real-world execution without treating failed actions as imitation targets.

7 Conclusion

We presented **PlayWAM**, which integrates controlled counterfactual play with branch-specific learning for World Action Models. By generating alternative futures from shared task states in simulation, our approach turns multiple PlayData into complementary supervision for physical dynamics and task-conditioned value. PlayWAM leverages this experience through selective action imitation and lightweight latent scoring, enabling single-round selection among parallel future–action candidates. Together, these designs extend play beyond trajectory diversity: the relationships among controlled alternatives provide a common foundation for learning physical consequences and selecting task-directed behavior.

AI USE STATEMENT

Generative AI tools were used to assist with drafting and polishing parts of the manuscript, retrieving and discovering related literature, and writing parts of the research code. All research ideas, methodological choices, and experimental designs were developed by the authors. The authors reviewed and revised all AI-assisted text, verified the references and technical claims, and tested the resulting code. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The main paper specifies the PlayWAM objectives, data conditions, and evaluation metrics. Appendix B details PlayData construction and interventions; Appendices C and D formalize the learning problem and implementation routes; and Appendix E reports benchmark composition, evaluation protocols, per-task results, and ablations. Together, these sections document the experimental evidence and procedures used in this work. All code, data, and benchmark assets will be released upon acceptance.

REFERENCES

Daniel S. Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 330–359, 2020. URL https://proceedings.mlr.press/v100/brown20a.html.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025. URL https://arxiv.org/abs/2506.18088.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://arxiv.org/abs/2303.04137.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1805.12114.

Ryan Hoque, Ajay Mandlekar, Caelan Garrett, Ken Goldberg, and Dieter Fox. IntervenGen: Interventional data generation for robust and data-efficient robot imitation learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. URL https://arxiv.org/abs/2405.01472.

Yixuan Huang, Novella Alvina, Mohanraj Devendran Shanthi, and Tucker Hermans. Fail2Progress: Learning from real-world robot failures with Stein variational inference. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 5581–5605. PMLR, 2025. URL https://proceedings.mlr.press/v305/huang25d.html.

Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A smooth sea never made a skilled sailor: Robust imitation via learning to search, 2025. URL https://arxiv.org/abs/2506.05294.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. CCIL: Continuity-based data augmentation for corrective imitation learning. In *International Conference on Learning Representations, 2024*, 2024. URL https://arxiv.org/abs/2310.12972.

Alexander Khazatsky et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024. URL https://arxiv.org/abs/2403.12945.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.06169.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. RoboMonkey: Scaling test-time sampling and verification for vision-language-action models. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3200–3217, 2025. URL https://proceedings.mlr.press/v305/kwok25a.html.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. DART: Noise injection for robust imitation learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 143–156, 2017. URL https://proceedings.mlr.press/v78/laskey17a.html.

Mingleyang Li, Yuran Wang, Yue Chen, Tianxing Chen, Jiaqi Liang, Zishun Shen, Haoran Lu, Ruihai Wu, and Hao Dong. GarmentPile++: Affordance-Driven Cluttered Garments Retrieval with Vision-Language Reasoning. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026a. URL https://arxiv.org/abs/2603.04158.

Runze Li, Hongyin Zhang, Junxi Jin, Qixin Zeng, Zifeng Zhuang, Yiqi Tang, Shangke Lyu, and Donglin Wang. World-value-action model: Implicit planning for vision-language-action systems, 2026b. URL https://arxiv.org/abs/2604.14732.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Jingbin Cai, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng Yan, Maoqing Yao, and Guanghui Ren. Genie Envisioner: A unified world foundation platform for robotic manipulation, 2025. URL https://arxiv.org/abs/2508.05635.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

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

Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. GarmentLab: A Unified Simulation and Benchmark for Garment Manipulation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2411.01200.

Haoran Lu, Songling Liu, Yue Chen, Guo Ye, Mutian Shen, Shuyang Yu, Yu Xiao, Shang Wu, Jiayi Wang, Jianshu Zhang, Jihai Zhao, Xiangtian Gui, Chuye Hong, Yuran Wang, Maojiang Su, Ruihai Wu, Zhaoran Wang, and Han Liu. MagicSim: A Unified Infrastructure for Executable Embodied Interaction. arXiv preprint arXiv:2606.17511, 2026a. URL https://arxiv.org/abs/2606.17511.

Haoran Lu, Mutian Shen, Shuyang Yu, Yu Xiao, Songling Liu, Jianshu Zhang, Shang Wu, Yue Chen, Guo Ye, Jiayi Wang, Zhaoran Wang, and Han Liu. AnnotateAnything: Automatic Annotation of 3D Assets for Robot Manipulation. arXiv preprint arXiv:2606.17446, 2026b. URL https://arxiv.org/abs/2606.17446.

Haoran Lu, Shang Wu, Songling Liu, Jianshu Zhang, Maojiang Su, Guo Ye, Chenwei Xu, Lie Lu, Pranav Maneriker, Fan Du, Zhaoran Wang, and Han Liu. Phys4D: Fine-Grained Physics-Consistent 4D Modeling from Video Diffusion. arXiv preprint arXiv:2603.03485, 2026c. URL https://arxiv.org/abs/2603.03485.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132, 2020. URL https://proceedings.mlr.press/v100/lynch20a.html.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of the 7th Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2310.17596.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2504.13059.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. arXiv preprint arXiv:2410.13816, 2024. URL https://arxiv.org/abs/2410.13816.

Chuanruo Ning, Ruihai Wu, Haoran Lu, Kaichun Mo, and Hao Dong. Where2Explore: Few-shot Affordance Learning for Unseen Novel Categories of Articulated Objects. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2309.07473.

Abby O'Neill et al. Open x-embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL https://arxiv.org/abs/2310.08864.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, 2018. doi: 10.1109/ICRA.2018.8460528. URL https://arxiv.org/abs/1710.06537.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019. URL https://arxiv.org/abs/1910.00177.

Physical Intelligence, Ali Amin, et al. $\pi^*_{0.6}$: A vla that learns from experience. arXiv preprint arXiv:2511.14759, 2025. URL https://arxiv.org/abs/2511.14759.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2026. URL https://arxiv.org/abs/2502.00622.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Yan Shen, Ruihai Wu, Yubin Ke, Xinyuan Song, Zeyi Li, Xiaoqi Li, Hongwei Fan, Haoran Lu, and Hao Dong. BiAssemble: Learning Collaborative Affordance for Bimanual Geometric Assembly. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2506.06221.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017. doi: 10.1109/IROS.2017.8202133. URL https://arxiv.org/abs/1703.06907.

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

Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play, 2023. URL https://arxiv.org/abs/2302.12422.

Ruihai Wu, Haozhe Chen, Mingtong Zhang, Haoran Lu, Yitong Li, and Yunzhu Li. Neural Dynamics Augmented Diffusion Policy. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13234–13241, 2025. doi: 10.1109/ICRA55743.2025.11128651. URL https://doi.org/10.1109/ICRA55743.2025.11128651.

Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06114.

Guo Ye, Zexi Zhang, Xu Zhao, Shang Wu, Haoran Lu, Shihan Lu, and Han Liu. Learning to Feel the Future: DreamTacVLA for Contact-Rich Manipulation. arXiv preprint arXiv:2512.23864, 2025. URL https://arxiv.org/abs/2512.23864.

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. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Tenny Yin, Zhiting Mei, Zhonghe Zheng, Miyu Yamane, David Wang, Jade Sceats, Samuel M. Bateman, Lihan Zha, Apurva Badithela, Ola Shorinwa, and Anirudha Majumdar. Playworld: Learning robot world models from autonomous play, 2026. URL https://arxiv.org/abs/2603.09030.

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

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, 2025a. URL https://arxiv.org/abs/2504.02792.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. IRASim: A fine-grained world model for robot manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025b. URL https://arxiv.org/abs/2406.14540.

A Appendix Overview

This appendix provides additional details on the simulation-generated PlayData, the formal training problem, the implementation of PlayWAM, and the complete experimental evaluation. The main paper presents the central idea and the headline results; here we make the data construction, supervision rules, and evaluation protocol sufficiently explicit to support interpretation and reproduction.

A.1 Scope of the Appendix

PlayData construction. Appendix B describes how we construct and annotate the five trajectory modes used by PlayWAM: Nominal, Recovery, No-Correction (NC), Counterfactual Intervention (CF), and Disturbance (DS). We focus on the TableTop and Mobile manipulation settings used in this work. In addition to the task subsets reported in the main benchmark, we provide the broader task inventory used during data generation and identify which tasks contribute to training and evaluation.

A.2 Organization of the Technical Details

Problem formulation. Appendix C expands the notation introduced in Section 3. It defines branch groups, intervention and outcome labels, temporal supervision masks, value targets, and the construction of future–action candidates. These definitions clarify how a trajectory can provide dynamics supervision without necessarily providing an action-imitation target.

Implementation. Appendix D gives implementation details for the data preprocessing pipeline, model components, branch-specific supervision, and single-round candidate selection.

Additional experiments. Appendix E reports the full evaluation protocol, task-level success rates, PlayData statistics, and additional ablations. In particular, the main-paper success rates are averages over the benchmark task groups; the appendix reports the corresponding results for each individual TableTop and Mobile task, together with analyses of branch supervision, value-based selection, and unseen disturbances.

Scope and terminology. Throughout the appendix, *intervention type* refers to how an off-nominal state is produced, whereas *branch type* refers to the trajectory mode used for learning. CF and DS trajectories may therefore end in either success or failure. NC and Recovery describe the continuation after an off-nominal state, and are not interchangeable with the binary task outcome.

B Simulation-Generated PlayData

B.1 Data Generation Scope

We generate PlayData in Isaac Sim using parallel physics-based rollouts. The collection procedure records visual observations, robot actions, robot state, task state, intervention events, and task outcomes for each episode. We focus on two embodiment settings used by this work: fixed-base TableTop manipulation and Mobile manipulation with a wheeled robot and an arm. The same collection interface supports nominal execution and controlled off-nominal execution, so the five PlayData modes described in the following subsections are recorded with a common trajectory format.

The collection inventory is broader than the 12-task benchmark reported in the main paper. This distinction is intentional. The full inventory is used to cover a range of interaction patterns during intervention-aware data generation, whereas the main benchmark uses a fixed subset for the aggregate quantitative comparisons in Section 6. In particular, the benchmark contains eight TableTop tasks and four Mobile tasks:

- **TableTop:** Grasp, PushT, RandomWave, Close Drawer, Open Drawer, Insert, PickPlace, and Pour;
- **Mobile:** Mobile Reach, Mobile Grasp, Mobile Open Drawer, and Mobile Close Drawer.

The benchmark subset is evaluated under nominal execution and controlled post-intervention conditions. The additional tasks in the collection inventory are used to document the scope of the intervention-enabled data generation pipeline and, where specified by an experiment, to provide training or auxiliary analysis data. They are not silently included in the 12-task benchmark averages.

B.1.1 Simulation and Embodiment Scope

Each episode is generated from a task specification containing a scene, a robot embodiment, an initial state distribution, a language or symbolic task goal, and a nominal execution procedure. TableTop episodes use a fixed-base arm in workspace-level manipulation tasks. Mobile episodes combine mobile-base motion with arm manipulation and therefore contain both navigation-related and end-effector actions. The collection records these actions in a unified representation while preserving embodiment-specific state fields for later analysis.

We use Isaac Sim as the simulation platform and build our data-collection infrastructure following the executable interaction framework described by MagicSim (Lu et al., 2026a). The resulting infrastructure integrates repeatable state initialization, task execution, intervention injection, retry-based recovery, and task-level annotation within a shared closed-loop environment. On top of this foundation, PlayData introduces the seeded branch-group construction described in Appendix B.2, where Nominal, Recovery, NC, CF, and DS trajectories are collected from a shared anchor state.

B.1.2 Task Families and Interaction Patterns

The task inventory spans several interaction patterns:

- **free-space motion:** reaching;
- **rigid-object manipulation:** grasping, placing, pick-and-place, and handover;
- **contact-rich manipulation:** pushing, sweeping, insertion, pouring, waving, and random play;
- **articulated-object interaction:** opening and closing drawers;
- **mobile manipulation:** mobile reaching, grasping, drawer opening/closing, and retraction-related manipulation.

This diversity allows the same PlayData construction principle to expose the model to changes in pose, contact, timing, force transmission, and execution reliability. The intervention mechanism is applied to all TableTop and Mobile tasks in the collection inventory; the exact intervention families and their task-specific parameterization are described in Appendix B.2.2.

B.1.3 Data Collection Modes

Every collected trajectory is assigned one branch type from the following five-mode vocabulary:

- *Nominal* records standard task execution without an explicit intervention;
- *Recovery* records a successful retry after an earlier failed attempt;
- *No-Correction (NC)* records the normal failed execution without a retry;
- *Counterfactual Intervention (CF)* changes a task-level action, target, contact, or timing variable;
- *Disturbance (DS)* perturbs the execution of an action at the control level.

Nominal, Recovery, and NC have fixed collection semantics: they represent a normal success, a successful retry after failure, and a normal failure without retry, respectively. CF and DS represent intervention-conditioned branches; their final task outcomes are recorded separately because an intervention may still result in either success or failure. The construction and labeling of these modes are specified in Appendix B.2.2.

B.1.4 Full Task Inventory and Benchmark Subset

Table 4 lists the intervention-enabled TableTop and Mobile task inventory used by the collection pipeline. A check mark in the last column identifies the tasks included in the 12-task main benchmark. Task names are written in functional form to make the distinction between the collection inventory and the evaluation subset explicit.

Table 4: **PlayData task inventory.** Intervention-aware collection covers all listed TableTop and Mobile tasks. The benchmark column identifies the eight TableTop and four Mobile tasks used for the main quantitative evaluation.

Setting	Task family	Task	Main benchmark
TableTop	Grasping	Grasp	✓
TableTop	Grasping	Bi-Grasp	
TableTop	Reaching	Reach	
TableTop	Reaching	Dual Reach	
TableTop	Contact	PushT	✓
TableTop	Contact	RandomPush	
TableTop	Contact	RandomPlay	
TableTop	Contact	Sweep	
TableTop	Contact	SweepSand	
TableTop	Contact	Wave	
TableTop	Contact	RandomWave	✓
TableTop	Placement	PickPlace	✓
TableTop	Placement	PickDrop	
TableTop	Placement	RandomPlace	
TableTop	Placement	Place	
TableTop	Insertion	Insert	✓
TableTop	Articulated object	Open Drawer	✓
TableTop	Articulated object	Close Drawer	✓
TableTop	Articulated object	Handover	
TableTop	Pouring	Pour	✓
TableTop	Garment object	Wash	
TableTop	Garment object	Fling	
TableTop	Garment object	Fold	
TableTop	Variant	XTrainer Grasp	
TableTop	Variant	XTrainer PickPlace	
Mobile	Reaching	Mobile Reach	✓
Mobile	Reaching	Mobile Dual Reach	
Mobile	Reaching	Mobile Retract Reach	
Mobile	Grasping	Mobile Grasp	✓
Mobile	Grasping	Mobile Dexterous Grasp	
Mobile	Articulated object	Mobile Open Drawer	✓
Mobile	Articulated object	Mobile Close Drawer	✓

Table 4 lists the complete collection inventory. Benchmark-specific task subsets, per-task trajectory counts, branch composition, and evaluation results are reported in Appendix E.

B.2 Simulation-Based PlayData Collection

Our basic data unit is a structured branch group rather than an isolated trajectory. For each sampled seed, we initialize an episode and execute a shared nominal prefix to a task-relevant branching point. We then snapshot the resulting simulator state, the *anchor state* defined in Sec. 3, and collect five controlled continuations from this same anchor. The seed fixes episode-level randomization, while the anchor fixes the physical and visual context at which the branches diverge. Consequently, differences between branches arise from their continuation protocols rather than from changes in the underlying scene state.

B.2.1 Shared Prefix and Anchor State

For branch group i, let z_i denote its collection seed. The seed determines episode-level randomized factors such as the initial robot configuration, object poses, appearance, camera configuration, and other task-specific scene parameters. Starting from this seeded episode, we execute a shared nominal prefix to a task-relevant branching point and snapshot the resulting physical state s_i^{anc} and context c_i^{anc}. All mode-specific continuations are then restored from this same anchor:

$$(z_i, s_i^{\text{anc}}, c_i^{\text{anc}}) \longrightarrow \left\{ \tau_i^{\text{Nominal}}, \tau_i^{\text{Recovery}}, \tau_i^{\text{NC}}, \tau_i^{\text{CF}}, \tau_i^{\text{DS}} \right\}. \tag{12}$$

Thus, branches within a group share the same physical state and task context at divergence, while differing only in their continuation protocol.

B.2.2 Five Trajectories per Branch Group

Each anchor produces a five-trajectory branch group:

$$\mathcal{G}_i = \left\{ \tau_i^{\text{Nominal}}, \tau_i^{\text{Recovery}}, \tau_i^{\text{NC}}, \tau_i^{\text{CF}}, \tau_i^{\text{DS}} \right\}. \tag{13}$$

The modes share the anchor state but differ in their execution protocol. Nominal is the standard successful execution. Recovery retries the task after an earlier failed attempt and records the successful retry. NC follows the normal execution without retry and records the resulting failure. CF changes a task-level target, action parameter, contact choice, or timing variable, while DS perturbs the execution of a control command.

The presence of a CF or DS intervention does not determine the task outcome; these branches may succeed or fail depending on the resulting physical state and subsequent execution. In contrast, the Nominal, Recovery, and NC modes are defined by their collection protocol and corresponding normal-success, retry-success, and no-retry-failure outcomes.

B.2.3 Shared Prefix and Continuation Point

All branches share the recorded prefix up to the anchor. At the branching point, the simulator state is cloned and independently restored before each mode-specific continuation:

$$\text{seeded episode} \;\rightarrow\; \text{shared prefix} \;\rightarrow\; \text{anchor state} \;\rightarrow\; \text{mode-specific continuation}. \tag{14}$$

B.2.4 Branch Packaging and Recorded Information

The five trajectories are stored as one branch group with shared-anchor metadata. Each trajectory retains its visual observations, actions, robot states, task states, branch mode, intervention descriptor, and final outcome. The group additionally stores the shared seed and the temporal locations of intervention, failure, and recovery events when they occur. This organization allows the same data unit to support three forms of supervision:

- individual trajectories supervise action-conditioned future dynamics;
- executable segments from successful nominal or corrective behavior supervise action generation;
- outcome differences between branches supervise task-conditioned value comparison.

B.2.5 Parallel Collection Procedure

We generate branch groups in parallel across independent simulator instances. For each seed, the collection procedure is:

1. initialize one randomized episode;
2. execute the shared prefix and snapshot the anchor state;
3. restore the anchor and construct the five mode-specific continuations;
4. record observations, actions, states, intervention events, and task outcomes for each continuation;

5. package the five trajectories and shared metadata as one branch group.

The resulting branch groups preserve both trajectory-level physical evidence and within-group outcome relations. The next subsections describe the construction of each mode and the corresponding labels in detail.

B.3 Mode-Specific PlayData Construction

The five trajectories in a branch group have fixed collection semantics. Nominal is the normal successful execution, Recovery is the successful retry after an earlier failure, and NC is the normal failure without retry. CF and DS apply controlled task-level and execution-level interventions, respectively. The following subsections describe how each mode is constructed and how it is used by PlayWAM.

Nominal, Recovery, and NC use the task-specific scripted expert without CF or DS. From the shared anchor, the expert samples feasible grasps or contacts and motion plans, so the same initial state can yield first-attempt success (Nominal) or failure through misses, slips, drops, planning or placement errors, or unmet success conditions. For Recovery–NC supervision, we retain pairs sharing that failed prefix and state: NC stops, whereas Recovery replans within its bounded retry budget and succeeds.

B.3.1 Nominal Trajectories

The Nominal trajectory executes the task from the shared anchor state without an explicit CF or DS intervention. The nominal controller follows the standard task procedure until the task succeeds. These trajectories provide the normal physical consequences and successful action demonstrations for the task. They are therefore used for future-dynamics supervision, action supervision, and positive value comparisons.

B.3.2 Recovery Trajectories

The Recovery branch continues from the shared anchor state. It first follows the task execution until an attempted execution fails, then retries the task with a corrective continuation. The final retry successfully completes the task. We retain both the preceding failed attempt and the successful retry, and record the transition between them so that the model can identify the action segment responsible for recovery.

Recovery trajectories provide three complementary signals: dynamics before and after the failed attempt, executable corrective actions, and a positive outcome reference for value learning. The recovery segment is consequently an action-positive segment even though the trajectory contains an earlier failure.

B.3.3 No-Correction (NC) Trajectories

The NC trajectory follows the normal task execution from the shared anchor state and records the resulting failure without retry. It preserves the failed continuation after the task becomes off-nominal, rather than replacing that continuation with a corrective behavior. NC therefore provides a direct failure counterpart to Recovery: both begin from the same anchor, while only the subsequent retry policy differs.

NC trajectories supervise action-conditioned future dynamics and provide negative outcome evidence for value learning. Their failure actions are not used as action-imitation targets because they describe what happened during the failed attempt rather than what the policy should execute.

B.3.4 Counterfactual Intervention (CF)

CF modifies a task-level variable while preserving the shared anchor state. Depending on the task, the intervention changes a target pose, contact choice, motion direction or magnitude, selected part, or action timing. The intervention is applied at the corresponding skill or action phase, after which the task continues under the modified condition.

Table 5: **Definitions and supervision roles of PlayData modes.** CF and DS may result in either success or failure; their action and value roles are therefore outcome-dependent.

Mode	Construction	Outcome	Dynamics	Action	Value
Nominal	Standard execution	Success	✓	✓	Positive
Recovery	Retry after failure	Success	✓	✓	Positive
NC	No retry after failure	Failure	✓	✗	Negative
CF	Task-level intervention	Success/Failure	✓	Conditional	Outcome-dependent
DS	Execution disturbance	Success/Failure	✓	Conditional	Outcome-dependent

CF is an intervention mode rather than an outcome label. A CF trajectory may successfully complete the task or terminate in failure. All recorded CF trajectories supervise action-conditioned future dynamics. Successful, executable CF segments additionally supervise action generation, whereas a failed CF continuation contributes to value comparison without becoming an action-imitation target.

B.3.5 Disturbance (DS)

DS perturbs the execution of an otherwise specified action after the shared anchor state has been established. The disturbance can be applied as action noise, motion noise, action dropout, or control delay, depending on the task and the selected execution phase. The task then continues under the perturbed control sequence.

As with CF, DS does not prescribe the final outcome: a disturbed trajectory may recover and succeed or may fail. Every recorded DS trajectory contributes to dynamics learning. Successful corrective segments provide action supervision, while failed DS continuations provide outcome evidence without asking the model to imitate the disturbance-induced failure action.

B.3.6 Supervision Roles of the Five Modes

Table 5 summarizes the construction semantics and the supervision role of each mode. "Conditional" action supervision means that only successful, executable segments from CF or DS trajectories are used as action targets. The value column describes the outcome role of a branch, not an additional input to the WAM.

B.3.7 Collection Scale and Branch Organization

For each task in the intervention-enabled TableTop and Mobile inventory, we collect 1,000 branch groups from independently sampled shared seeds. Each branch group contains one trajectory for each of the five modes, yielding 5,000 mode-specific trajectories per task. The inventory contains 25 TableTop tasks and 7 Mobile tasks, for 32 tasks in total. Together, the collection contains 32,000 branch groups, 160,000 mode-specific trajectories, and over 1,000 hours of annotated PlayData.

The five trajectories belonging to one branch group are kept together whenever forming a data split. This preserves the shared-anchor relation for value learning and prevents trajectories from one branch group from being distributed across different data subsets.

C Detailed Problem Formulation

C.1 Context, State, and Branch Groups

This subsection fixes the notation used throughout the remainder of the appendix. PlayWAM does not treat an episode as an isolated sequence: the training unit is a set of trajectories that share one anchor state and then follow different continuation protocols. This organization allows the model to compare nominal behavior, recovery, failure, and intervened behavior under the same scene configuration.

C.1.1 Task Context

At decision time t, the task context is denoted by

$$c_t = \big(o_{t-L:t},\, q_t,\, g\big), \tag{15}$$

where $o_{t-L:t}$ is a window of the most recent visual observations, q_t contains the robot state available to the policy, and g is the task goal or instruction. For a Mobile task, q_t includes both base and arm state; for a TableTop task it contains the corresponding fixed-base robot state.

C.1.2 Shared Anchor State

Let i index a branch group and let z_i denote its collection seed. The seed determines the randomized episode configuration. We execute a shared prefix to a task-relevant branching point and denote the resulting physical state and task context by s_i^{anc} and c_i^{anc}, respectively. The anchor of branch group i is

$$x_i^{\mathrm{anc}} = (z_i, s_i^{\mathrm{anc}}, c_i^{\mathrm{anc}}). \tag{16}$$

All mode-specific continuations in the group are restored from the same x_i^{anc}. We denote the robot-state component of c_i^{anc} by q_i^{anc}.

C.1.3 Trajectory Representation

For a mode $m \in \mathcal{M}$, where

$$\mathcal{M} = \{\mathrm{Nominal}, \mathrm{Recovery}, \mathrm{NC}, \mathrm{CF}, \mathrm{DS}\}, \tag{17}$$

we represent the resulting trajectory as

$$\tau_i^m = \big\{(o_t, q_t, a_t, s_{t+1}, r_t, y_t)\big\}_{t=0}^{T_i^m - 1}. \tag{18}$$

Here a_t is the executed action, s_{t+1} is the next physical state, r_t is the step-level task-progress signal, and y_t stores the task or event labels associated with that transition. The final outcome is denoted by $Y_i^m \in \{0, 1\}$, with $Y_i^m = 1$ indicating task success. The mode label m and the outcome Y_i^m are recorded separately: in particular, CF and DS identify how a trajectory was generated, while their outcomes can be either success or failure.

C.1.4 Branch Group Definition

The structured training unit is the five-trajectory branch group

$$\mathcal{G}_i = \Big(x_i^{\mathrm{anc}},\, \{\tau_i^m\}_{m \in \mathcal{M}}\Big). \tag{19}$$

Each τ_i^m is a mode-specific continuation from the same anchor, which defines the within-group comparison. Nominal and Recovery provide successful action-bearing examples, NC provides a no-retry failure reference, and CF/DS provide intervened trajectories whose supervision depends on the recorded outcome. This grouping is used later to construct temporal, value, and candidate relations.

C.2 Branch Types and Outcomes

Each trajectory in a branch group defines one *branch*: a continuation from the group's shared anchor state under a particular execution mode. A branch type describes how that continuation was constructed, whereas its outcome describes whether the task was completed. Keeping these two attributes separate is important for PlayWAM: an intervention identifies an alternative physical continuation, but it does not by itself determine whether that continuation is useful or successful.

C.2.1 Mode Semantics

For each branch group, every branch has a mode variable $m \in \mathcal{M}$, where

$$\mathcal{M} = \{\mathrm{Nominal}, \mathrm{Recovery}, \mathrm{NC}, \mathrm{CF}, \mathrm{DS}\}. \tag{20}$$

The five modes have the following semantics:

- *Nominal* is the standard task execution and is collected as a successful reference trajectory;
- *Recovery* follows an earlier failed attempt with a retry and records a successful retry;
- *NC* follows the normal execution after the off-nominal event without retry and records the resulting failure;
- *CF* changes a task-level choice, such as a target, contact, action parameter, or timing variable;
- *DS* perturbs the execution of an otherwise specified control sequence.

The first three modes therefore have fixed collection semantics: normal success, retry success, and no-retry failure. CF and DS are intervention modes rather than outcome labels. Either mode can produce a successful or a failed continuation, depending on the resulting physical state and the subsequent actions.

C.2.2 Trajectory Outcomes

Let $Y_i^m \in \{0, 1\}$ denote the final task outcome of trajectory τ_i^m, where one indicates task success. Under the collection semantics above,

$$Y_i^{\text{Nominal}} = 1, \qquad Y_i^{\text{Recovery}} = 1, \qquad Y_i^{\text{NC}} = 0, \tag{21}$$

while

$$Y_i^{\text{CF}},\, Y_i^{\text{DS}} \in \{0, 1\}. \tag{22}$$

These constraints define branch construction; the outcome variable remains explicitly recorded for every branch. In particular, the same intervention type can appear in both successful and failed continuations across different shared physical states.

C.2.3 Branch-Level Outcome Comparisons

The outcome structure of branch group $\mathcal{G}_i$ can be summarized by

$$\mathbf{Y}_i = \big(Y_i^{\text{Nominal}}, Y_i^{\text{Recovery}}, Y_i^{\text{NC}}, Y_i^{\text{CF}}, Y_i^{\text{DS}}\big). \tag{23}$$

Because all entries in this vector belong to the same branch group and share the same anchor state, they can be used to compare alternative continuations without conflating outcome differences with changes in the anchor context. For example, a successful Recovery branch and an NC branch form a local contrast between retry and no-retry continuations after an off-nominal state. Similarly, successful and failed CF or DS branches provide direct comparisons of the task utility of intervened futures.

C.2.4 Outcome-Dependent Supervision Eligibility

All recorded trajectories provide evidence about action-conditioned physical dynamics. Action generation uses only successful, executable behavior: Nominal and successful Recovery segments are eligible, as are executable, task-aligned segments from successful CF/DS continuations. Failure-causing segments are retained as physical observations but are not imitation targets.

For value learning, the relevant signal is the relative utility of comparable branches that start from the same shared physical state. Thus, a failed CF or DS branch can be the lower-utility member of a within-group pair against a successful continuation. The value target is therefore an outcome-dependent ordering rather than a mode-dependent rule: CF and DS are not intrinsically positive or negative, and their role is determined by the recorded outcome and the matched within-group comparison.

C.3 Temporal Supervision Labels

Branch outcomes alone do not identify which actions within a trajectory should be imitated. A Recovery branch, for example, contains both an unsuccessful attempt and a successful corrective continuation. We therefore attach time-aligned event labels and construct separate masks for physical-dynamics and action-generation supervision.

C.3.1 Event-Aligned Branch Timeline

For branch τ_i^m, we denote the intervention, failure, recovery-start, and recovery-end times by

$$t^m_{i,\mathrm{int}}, \qquad t^m_{i,\mathrm{fail}}, \qquad t^m_{i,\mathrm{rec}}, \qquad t^m_{i,\mathrm{end}}, \tag{24}$$

whenever the corresponding event is present. Nominal branches have no intervention, failure, or recovery event. NC branches contain a failure without a recovery. Recovery branches contain the failed attempt and the subsequent corrective interval. CF and DS branches record the intervention time and any failure or recovery events that occur afterward. These labels describe the observed execution timeline; they do not change the branch type or determine its final outcome.

Let $v^m_{i,t} \in \{0,1\}$ indicate that the transition at time t is valid and has the observations, action, and next state required for training. We also define $e^m_{i,t} \in \{0,1\}$ as an executable-segment indicator. It is one when the recorded action belongs to a task-aligned successful behavior, and zero for a failure-causing interval or an externally injected disturbance that the policy should not reproduce.

C.3.2 Dynamics Supervision Mask

The dynamics mask retains every recorded physical transition:

$$m^{\mathrm{dyn},m}_{i,t} = v^m_{i,t}. \tag{25}$$

Consequently, recorded Nominal, Recovery, NC, CF, and DS branches all contribute to learning action-conditioned future dynamics. A transition remains useful for this objective even when its action should not be imitated, because it still records how the scene evolves under that action or disturbance.

C.3.3 Action Supervision Mask

The action mask selects successful, executable behavior rather than selecting entire branch types. For Nominal, NC, and Recovery branches, it is defined as

$$m^{\mathrm{act},m}_{i,t} = \begin{cases} v^m_{i,t}, & m = \mathrm{Nominal}, \\ 0, & m = \mathrm{NC}, \\ v^m_{i,t}, & m = \mathrm{Recovery},\ t < t^m_{i,\mathrm{fail}}, \\ 0, & m = \mathrm{Recovery},\ t^m_{i,\mathrm{fail}} \le t < t^m_{i,\mathrm{rec}}, \\ v^m_{i,t}, & m = \mathrm{Recovery},\ t \ge t^m_{i,\mathrm{rec}}. \end{cases} \tag{26}$$

Thus, the normal prefix and successful corrective suffix of a Recovery branch supervise action generation, while the failure-causing interval is excluded. NC is retained for dynamics and value supervision but provides no action-imitation target.

For intervention branches, action eligibility additionally depends on the recorded outcome and segment quality:

$$m^{\mathrm{act},m}_{i,t} = v^m_{i,t}\, Y^m_i\, e^m_{i,t}, \qquad m \in \{\mathrm{CF}, \mathrm{DS}\}. \tag{27}$$

A successful CF or DS branch therefore provides action supervision on its executable, task-aligned segments. Failed continuations, failure-causing segments, and the externally injected component of a disturbance have zero action mask. This implements the principle that supervision follows the observed outcome and segment quality rather than treating intervention modes as intrinsically positive or negative.

C.3.4 Supervision over Sampled Windows

A sampled training window may cross an intervention, failure, or recovery boundary. We preserve the time-indexed masks inside the window instead of assigning one supervision label to the entire trajectory. The dynamics and action objectives therefore aggregate only their eligible time steps:

$$\mathcal{L}^{(i,m)}_{\mathrm{dyn}} \propto \sum_t m^{\mathrm{dyn},m}_{i,t} \ell^{\mathrm{dyn}}_{i,t}, \qquad \mathcal{L}^{(i,m)}_{\mathrm{act}} \propto \sum_t m^{\mathrm{act},m}_{i,t} \ell^{\mathrm{act}}_{i,t}. \tag{28}$$

The exact loss normalization and optimization schedule are given in Appendix D.3. Value supervision is defined at the branch-comparison level rather than by these temporal action masks and is detailed next.

C.4 Value Targets and Pair Construction

The binary outcome Y_i^m indicates whether a branch completes the task, but it does not distinguish different degrees of progress among branches with the same outcome. PlayWAM therefore associates each branch with a scalar task utility and learns value primarily from reliable within-group orderings.

C.4.1 Outcome and Branch Utility

Let R_i^m denote the recorded task return of branch τ_i^m. We map it to a task-normalized utility

$$G_i^m = \mathrm{NormalizeTask}(R_i^m), \qquad G_i^m \in [0,1]. \tag{29}$$

The return combines the final task outcome with observed task progress. Successful branches receive higher utility than failed branches, while progress orders unsuccessful continuations. Thus, Y_i^m remains the binary success annotation and G_i^m is the scalar target used for value comparison.

C.4.2 Within-Group Comparable Pairs

Value comparisons are constructed only between branches in the same branch group. Since these branches start from the same anchor state, their utility difference reflects the quality of their continuations rather than a change in context. Given a minimum reliable utility gap $\epsilon_v > 0$, we define

$$\mathcal{P}_i = \left\{ (m^+, m^-) \;\middle|\; m^+, m^- \in \mathcal{M}, \quad G_i^{m^+} - G_i^{m^-} > \epsilon_v \right\}. \tag{30}$$

The first member of each pair is the preferred continuation. Branches from different groups are not paired, and branches whose utilities differ by less than ϵ_v do not receive a forced ordering. This is particularly important when, for example, a Nominal branch and a Recovery branch both succeed with nearly identical utility.

C.4.3 Canonical Within-Group Comparisons

The pair definition is utility based rather than mode based. It naturally produces several recurring comparisons, including

$$\text{Recovery} \succ \text{NC}, \tag{31}$$

and, more generally,

$$\text{successful continuation} \succ \text{lower-utility failed continuation}. \tag{32}$$

Depending on the observed outcomes within a group, the latter can compare a Nominal or Recovery branch against a failed CF, DS, or NC branch; it can also compare a successful CF or DS branch against another lower-utility continuation. These are examples of the relation induced by G_i^m, not hard-coded priorities between mode names. In particular, CF and DS branches can occur on either side of a preference pair according to their recorded task utility.

C.4.4 Value Ordering Target

Let h_i^m denote the contextualized, completed future–action representation of branch τ_i^m. The value head predicts

$$\hat{v}_i^m = V_\phi(\mathrm{sg}(h_i^m)), \tag{33}$$

where h_i^m already encodes the anchor context through WAM conditioning and sg stops value gradients from updating the shared WAM. For every $(m^+, m^-) \in \mathcal{P}_i$, the desired ordering is

$$\hat{v}_i^{m^+} > \hat{v}_i^{m^-}. \tag{34}$$

Equivalently, the preference probability can be written as

$$p_\phi(m^+ \succ m^-) = \sigma\left(\frac{\hat{v}_i^{m^+} - \hat{v}_i^{m^-}}{\tau_v} \right), \tag{35}$$

where τ_v is a ranking temperature. Pairwise ordering is the primary value signal. Regression toward G_i^m provides auxiliary absolute calibration, allowing scores to remain interpretable across decisions. The corresponding ranking and calibration objectives are given in Sec. 5.2.

C.5 Candidate Generation and Selection

At inference time, PlayWAM applies the relations learned from PlayData to a set of generated future–action candidates. All candidates share the current observation context and task goal, but represent different possible continuations. They are generated in parallel, scored after completion, and used for one-shot action selection.

C.5.1 Parallel Future–Action Candidates

Given the current context

$$c_t = (o_{t-L:t}, q_t, g), \tag{36}$$

we sample independent latent noise variables $\{\epsilon_t^b\}_{b=1}^B$ and use the shared WAM to generate

$$\hat{\xi}_t^b = F_\theta(c_t, \epsilon_t^b) = \left(\hat{o}_{t+1:t+H}^b, \hat{a}_{t:t+H-1}^b\right), \qquad b = 1, \dots, B. \tag{37}$$

The candidates use the same real visual history, robot state, and task goal. Their diversity comes from the independently sampled latent variables. Each candidate jointly specifies a possible future observation sequence and its corresponding action sequence over horizon H. The B candidates are generated as one batched set rather than through a sequence of value-guided updates.

C.5.2 Completed Candidate Representation

After a candidate has completed generation, we obtain a compact latent representation

$$h_t^b = \mathrm{Pool}\left(H_{z,t}^b, H_{a,t}^b\right), \tag{38}$$

where the video and action hidden states have already been conditioned on c_t. Thus, h_t^b summarizes the predicted future, action continuation, and task context without a separate context input to the value head. The value head then produces

$$\hat{v}_t^b = V_\phi\left(\mathrm{sg}(h_t^b)\right). \tag{39}$$

Scoring the completed representation allows the value estimate to depend on the predicted consequence of the candidate action sequence. The sg operator preserves the separation between candidate generation and value estimation. Candidate futures remain in latent form for scoring and do not need to be decoded into pixel-space videos.

C.5.3 Single-Round Value Selection

The selected candidate is

$$b^* = \arg\max_{b \in \{1,\dots,B\}} \hat{v}_t^b. \tag{40}$$

All candidates are fully generated once and scored once. Selection operates over this single batched candidate set.

C.5.4 Receding-Horizon Execution

Rather than executing the full predicted horizon, the controller applies only the first h_{exec} actions of the selected candidate:

$$a_t^{\text{exec}} = \hat{a}_{t:t+h_{\text{exec}}-1}^{b^*}, \qquad h_{\text{exec}} \le H. \tag{41}$$

The resulting real observation is appended to the context, after which a new parallel candidate set is generated for the next decision. Predictions are therefore used to compare possible futures, while closed-loop control remains grounded in observations from the environment. Sec. 6.1 reports the candidate budget and defines inference efficiency as the action-chunk generation rate.

D Implementation Details

D.1 Data Preprocessing and Branch-Group Batching

Appendix B describes how the five PlayData trajectories are collected. This section describes the subsequent transformation from recorded trajectories to the model inputs used by PlayWAM. The

preprocessing pipeline retains the branch-group structure throughout: visual observations, actions, robot states, event annotations, and task outcomes remain associated with the shared anchor state from which their continuations were collected.

D.1.1 CANONICAL TRAJECTORY REPRESENTATION

Each recorded branch is converted into a canonical multimodal record. The record contains an RGB observation sequence, a robot-action sequence, proprioceptive states, the task instruction, the branch type, the final success label, and a task-normalized scalar utility. It also stores the intervention, failure, and recovery timestamps when they are present. These timestamps are used for temporal window sampling and supervision masks; they are not provided to the model as branch-type identifiers.

Table 6 summarizes the fields retained for training. The branch-group identifier is attached to the complete set of continuations rather than to an individual trajectory. Consequently, it can be used to construct matched training batches and within-group value pairs without exposing the model to a mode label at inference time.

Table 6: **Canonical PlayData training record.** Event and branch metadata control sampling and loss construction, but are not used as test-time mode inputs.

Field	Granularity	Training role
Visual observations	Per frame	Context frames and future-video targets
Robot actions	Per control step	Action targets or conditions for dynamics prediction
Robot state	Per control step	Current-state conditioning
Event timestamps	Per branch	Event-aware window sampling and temporal masks
Success and utility	Per branch	Outcome labels and value-pair construction
Branch-group identity	Per group	Matched sampling, group-preserving splits, and comparisons

We use a common action layout across tasks within an embodiment. Fixed-base TableTop training uses an eight-dimensional action consisting of end-effector pose and gripper control. Mobile training uses a twelve-dimensional layout that additionally represents mobile-base motion. Quaternion signs are canonicalized before normalization to remove the equivalent q and $-q$ representations. Action ranges are estimated jointly across the training tasks of the same embodiment, so an identical physical command has a consistent normalized representation across tasks. Task utilities, in contrast, are normalized per task because progress and return scales are task-specific.

D.1.2 VISUAL, ACTION, AND TEMPORAL WINDOWS

RGB frames are resized to 256×256, scaled to the model input range, and then encoded by the pretrained video tokenizer. A training window contains four context frames and nine future frames sampled with a temporal stride of four control steps. The corresponding action target contains 36 consecutive control steps, aligning the future visual horizon with the higher-frequency robot commands. Robot state is taken at the final context step and provides the current proprioceptive condition used by both supervision routes.

Uniform temporal sampling alone would overrepresent uneventful prefixes in long trajectories and could omit the state changes that distinguish the five branch types. We therefore use mixed window sampling. For branches with an intervention, failure, or recovery timestamp, half of the windows are centered on one recorded event and the remaining windows are drawn uniformly from the valid trajectory range. Branches without an event use uniform sampling. This procedure exposes the model both to ordinary task execution and to the local physical consequences of interventions, failures, and corrective behavior.

For every sampled window, the video and action sequences are cropped using the same temporal origin. Time-indexed action masks are cropped with the action window, ensuring that a window crossing a failure or recovery boundary retains the segment-specific eligibility defined in Appendix C.3.3. Missing event-centered segments are not synthesized or padded with supervision; only recorded valid steps contribute to their corresponding objectives.

D.1.3 Group-Preserving Data Allocation and Model Inputs

The PlayBench training sets follow the composition in Appendix E.3. Nominal uses 1,000 successful Nominal trajectories per task. Play-Full uses all five trajectories from each of 200 complete branch groups, and Play-Success retains the naturally occurring successful trajectories from those same groups. Validation assignment is performed at the branch-group level so that all continuations from one shared anchor remain together. Benchmark evaluation instead uses disjoint simulator seeds and held-out scene configurations, as described in Appendix E.3.1.

The same preprocessed records support the two optimization stages described in the main paper. During WAM adaptation, a batch contains executable branches and dynamics-only branches selected from the branch-group records, together with their time-indexed supervision masks. During value learning, a batch retains comparable continuations from the same branch group and provides their utilities for within-group pair construction. The data loader preserves a leading branch dimension in both cases, allowing the trainer to distinguish branches while sharing the task instruction and group identity.

For branch group $\mathcal{G}_j$, the resulting model-ready record can be written as

$$\mathcal{X}_j = \left\{ O_j^b, A_j^b, q_j^b, m_{\mathrm{dyn}}^{j,b}, m_{\mathrm{act}}^{j,b}, m_{\mathrm{val}}^{j,b}, G_j^b \right\}_{b \in \mathcal{G}_j}, \tag{42}$$

where O_j^b and A_j^b are the aligned video and action windows, q_j^b is the current robot state, the three masks control eligibility for dynamics, action, and value supervision, and G_j^b is the task-normalized branch utility. The masks determine which losses consume each field; their concrete routing through the two WAM update paths is detailed in Appendix D.3.

D.2 Model Architecture

PlayWAM instantiates the formulation in Sec. 5 with three components: a pretrained video world-model backbone augmented with an action expert, an action-conditioned dynamics path, and a lightweight trajectory-value head. The resulting WAM jointly models future observations and actions, the dynamics path learns the physical consequences of recorded failure actions without treating them as behavioral targets, and the value head evaluates each completed branch. All three components operate on the model-ready branch records defined in Appendix D.1; their supervision is routed by the masks described in Appendix D.3.

D.2.1 World–Action Backbone and Parallel Branch Processing

The video world-model backbone is initialized from the GE-Base-fast v0.1 checkpoint of Genie-Envisioner, while the video tokenizer, text encoder, and VAE use the pretrained LTX-Video components. Because GE-Base does not contain an action expert, the action modules are newly initialized. The action-to-video adapters use zero-initialized output projections, preserving the pretrained world-model behavior when they are attached. TableTop and Mobile models are trained as separate checkpoints with embodiment-specific action dimensions, but both originate from the same GE-Base initialization.

Visual context and future RGB frames are encoded into compact video latents by the pretrained video tokenizer, while the task instruction is mapped to text context tokens. The dedicated action expert embeds continuous robot commands and predicts their flow-matching velocity. The video and action streams retain modality-specific input and output projections while exchanging information in the context-conditioned Transformer. The resulting network predicts the future-video and action flows in Eq. 5, together with the hidden representation later used to evaluate the branch.

The visual history, task instruction, and current robot state define the shared decision context. Future video and action sequences vary across branches, but the encoders and Transformer parameters are shared across them. During parallel generation, the branch axis is folded into the batch axis. Given an encoded context C_t, branch b starts from independent video–action noise $(\epsilon_z^b, \epsilon_a^b)$ at $\tau = 1$ and follows the reverse schedule to $\tau = 0$. Thus, candidate diversity arises from independent initial noise, while batched computation reuses the context and model parameters. Each completed branch retains its own predicted video features, action features, and action sequence. Mode and outcome

annotations are used for data sampling and supervision routing rather than as branch-generation inputs.

D.2.2 Action-Conditioned Dynamics Path

For a dynamics-only branch, the recorded action specifies the behavior whose physical consequence should be modeled. We encode this clean action sequence at flow time zero, detach its representation, and inject it into the video stream through lightweight action-to-video adapters. The adapters are inserted periodically between video Transformer blocks and use zero-initialized output projections, preserving the pretrained WAM at initialization while allowing the visual dynamics to become explicitly action-conditioned during training.

This path implements an explicitly action-conditioned dynamics model,

$$p_\theta\big(z_{t+1:t+H} \mid C_t, a^{\text{recorded}}_{t:t+H-1}\big), \tag{43}$$

rather than predicting future video from context alone. During this update, the recorded action is consumed as a condition and the loss is applied only to the video flow. A failed segment can therefore teach the model what follows from that behavior without training the action generator to reproduce it. At joint inference, the same adapters remain active and receive the current denoising action tokens, keeping each predicted future conditioned on its candidate action throughout generation.

D.2.3 Detached Trajectory-Value Head

The value module scores a completed imagined branch rather than the current state alone. Let H^b_z and H^b_a denote the final video and action hidden tokens of branch b. We mean-pool the two modalities, project them into a common hidden space, and add the flow-time embedding before predicting a scalar value:

$$\begin{aligned} \bar{h}^b_z &= \mathrm{MeanPool}(H^b_z), & \bar{h}^b_a &= \mathrm{MeanPool}(H^b_a), \\ h^b_\tau &= P_z\,\mathrm{sg}(\bar{h}^b_z) + P_a\,\mathrm{sg}(\bar{h}^b_a) + E_\tau(\tau), & \hat{v}^b &= \mathrm{MLP}_\phi(h^b_\tau). \end{aligned} \tag{44}$$

The WAM hidden states have already attended to the shared context $C_t = (o_{t-L:t}, q_t, g)$, so task and robot-state information is encoded in both pooled modalities rather than passed through separate value-head encoders. At inference, the head scores completed branches at $\tau = 0$. Because the readout operates directly on hidden tokens, candidate RGB videos do not need to be decoded before selection. The stop-gradient operator isolates the generator from the value-learning gradients, so branch outcomes supervise the value head without altering the video or action representation through this objective. The contextualized future–action representation identifies whether the predicted consequence is useful for the present decision, enabling the head to rank successful and failed consequences within the same branch group. The corresponding ranking objective is given in Sec. 5.2. After Stage 1, we load the adapted WAM checkpoint, freeze all generative parameters, and optimize only the detached trajectory-value head in Stage 2. Stage 2 encodes recorded video–action windows through the frozen joint path. At inference, generated future–action candidates follow the same joint path before scoring, so training and inference use the same representation route. For singleton Nominal data, no within-group preference pair is available and the value head uses calibration alone; Play-Full additionally supplies same-anchor ranking pairs.

D.3 Branch-Specific Supervision

The masks introduced in Appendix C.3 are applied at the segment level rather than assigning one optimization role to an entire trajectory. This is essential for Recovery, CF, and DS branches, which may contain both executable behavior and failure-causing or externally perturbed intervals. We implement the resulting assignment with two complementary WAM update routes. The branch-level eligibility variable $m_{j,b}$ in the main-paper objective is compact notation for this routing decision; in implementation, it is expanded to the time-indexed mask $m^{\text{act},m}_{i,t}$ so that mixed windows supervise only their successful, executable steps.

D.3.1 Executable and Dynamics-Only Update Routes

Table 7 summarizes the two routes. In the *executable joint route*, both video and action targets are independently noised at the sampled flow time. The model predicts both flow fields, and the

video and action objectives jointly update the world–action backbone. This route is used only for task-aligned behavior that the policy should reproduce.

In the *dynamics-only route*, the future-video target is noised as usual, whereas the recorded action is encoded at flow time zero and detached before being supplied through the action-conditioned dynamics path. The update optimizes only future-video prediction. Action-specific parameters are frozen for this route, while the shared world path and action-to-video adapters learn the observed physical consequence. Consequently, a failure action remains available as an explanatory condition without becoming an action target.

Table 7: **Implementation of the two supervision routes.** The routes share the world backbone but differ in how actions enter the model and which outputs receive supervision.

Route	Action input	Objective	Updated components
Executable joint	Noised action latent α_τ	Video flow + action flow	Shared world–action backbone and action expert
Dynamics only	Clean, detached action $\mathrm{sg}(\alpha)$	Video flow only	Shared world path and action-to-video adapters

Within each branch group, executable and dynamics-only contributions are normalized separately as in Eq. 7. This keeps the scale of successful action learning stable when the number or duration of failure segments varies across branch groups. We interleave the two update routes during WAM adaptation, so the model acquires broader physical dynamics while continually retaining task-effective action supervision.

D.3.2 Segment-Level Routing of the Five Modes

Table 8 instantiates the temporal masks for the five PlayData modes. Nominal and NC have a single route throughout their valid windows. Recovery is split at the recorded failure and recovery events: its task-aligned prefix and corrective suffix use the executable route, while the failure-causing interval uses the dynamics-only route. CF and DS are routed according to their recorded outcome and segment quality rather than their mode name.

Table 8: **Segment-level supervision routing.** Value roles are assigned at the branch level; video and action roles follow the time-aligned segment masks.

Branch	Executable joint route	Dynamics-only route	Value role
Nominal	All successful execution segments	—	Positive
Recovery	Task-aligned prefix and corrective suffix	Failure-causing interval	Positive
NC	—	All failure segments	Negative
CF success	Executable, task-aligned segments	Non-executable intervention or failure intervals	Higher utility
CF failure	—	All recorded segments	Lower utility
DS success	Executable continuation and corrective segments	Externally injected disturbance and failure intervals	Higher utility
DS failure	—	All recorded segments	Lower utility

For a successful CF or DS branch, action supervision is therefore restricted to segments satisfying $m_{i,t}^{\mathrm{act},m} = 1$. In particular, an externally injected DS command remains dynamics-only even if the subsequent continuation recovers and completes the task. Value learning instead uses the utility of the complete branch: successful CF/DS branches can serve as preferred continuations, whereas failed CF/DS and NC branches provide lower-utility comparisons. The same mode can consequently occupy either side of a value pair when supported by its observed outcome and utility.

D.3.3 Failure-Aware World Modeling and Behavioral Preservation

The quality of action generation in a WAM depends on the quality of its anticipated visual future; inaccurate or ambiguous video prediction can propagate to the action stream (Ye et al., 2026). Failure data make this prediction problem broader by adding collisions, missed contacts, ineffective motions, and disturbed transitions that are absent from successful demonstrations. These outcomes are nevertheless important physical evidence: a model trained only around successful executions has little basis for predicting what follows when a candidate action begins to deviate from the task.

PlayWAM organizes this evidence through the supervision routes above. First, the recorded action conditions each failure future, associating the observed consequence with the behavior that produced it. Second, the dynamics-only route prevents failure-causing actions from entering the imitation objective, while interleaved executable updates preserve successful action generation. Third, within-group value learning uses the resulting successful and failed consequences to distinguish physical plausibility from task desirability. At inference, a candidate may represent a physically valid but unsuccessful future; the value head is trained to rank it below a successful or corrective continuation from the same decision context. Failure trajectories therefore expand the modeled consequence distribution, while action routing and value selection determine which parts of that distribution shape executed behavior.

D.4 Training and Inference Configuration

Table 9 summarizes the principal optimization and inference settings used in our experiments.

Table 9: **PlayWAM training and inference configuration.**

Setting	Value
Stage 1 optimizer / learning rate	AdamW / 1×10^{-4}
Stage 1 global batch / steps	128 branch groups / 80000 (TableTop), 40000 (Mobile)
Stage 1 schedule / warmup	cosine / 1000 steps
Stage 2 optimizer / learning rate	AdamW / 5×10^{-5}
Stage 2 global batch / steps	8 branch groups / 8k (TableTop), 4k (Mobile)
Stage 2 schedule / warmup	constant / 10%
Loss weights $(\lambda_a, \lambda_{\mathrm{dyn}}, \lambda_{\mathrm{cal}})$	$(1.0, 1.0, 0.2)$
Ranking temperature τ_v / utility margin ϵ_v	$0.2/0.1$
Inference (H, S, B)	$(36, 10, 4)$
Precision / hardware	BF16 / NVIDIA A800 (8 GPUs Stage 1; 8 GPUs Stage 2)

E Additional Experiments

E.1 Benchmark Tasks and Intervention Protocols

This section specifies the task-level evaluation protocols that underlie the aggregate results in the main paper. PlayBench and RoboTwin-Play evaluate closed-loop behavior both with and without controlled interventions, whereas LIBERO retains its standard evaluation protocol without custom interventions. The intervention definitions below describe observable changes applied during evaluation and do not encode the final outcome: an intervened episode may still succeed or fail according to the original task success criterion.

All reported inference rates are measured locally on a single NVIDIA A800 GPU under each benchmark's native inference configuration. For LIBERO, only WAV's success rates are taken from its original paper; its Hz is measured locally under this timing protocol.

E.1.1 Evaluation Conditions

We use three evaluation conditions for PlayBench and RoboTwin-Play. *Nominal* executes the task without an additional test-time intervention. *Post-CF* applies one task-specific counterfactual change at an eligible action-chunk boundary, after which the policy observes the altered state and replans.

Post-DS perturbs action execution during one complete action chunk, after which the policy receives the resulting observation and resumes its standard closed-loop cycle. These evaluation conditions are distinct from the five PlayData branch types: they describe the test environment presented to the policy rather than the supervision role of a training trajectory.

For each task, the intervention family, magnitude, and trigger phase are drawn from a fixed task-specific specification. The environment's original success function remains unchanged across Nominal, Post-CF, and Post-DS. This allows the reported success rate to measure whether the policy completes the same task after a controlled change, rather than treating intervention occurrence as failure by definition.

E.1.2 PLAYBENCH TASKS AND SUCCESS CRITERIA

PlayBench contains eight fixed-base TableTop tasks and four Mobile manipulation tasks in Isaac Sim. Table 10 summarizes the task objective and success criterion used for evaluation.

Table 10: **PlayBench task definitions.** The benchmark contains eight TableTop tasks and four Mobile manipulation tasks. All three evaluation conditions use the same task-specific success criterion.

Setting	Task	Objective	Success criterion
TableTop	Grasp	Grasp and lift the target object	The target is securely grasped and lifted to the required height.
TableTop	PushT	Push the object to its target	The object reaches the designated target region within the position tolerance.
TableTop	RandomWave	Grasp and wave the object	The object remains grasped while the required waving motion is completed.
TableTop	Open Drawer	Open an articulated drawer	The drawer joint reaches the task-specific open threshold.
TableTop	Close Drawer	Close an articulated drawer	The drawer joint reaches the task-specific closed threshold.
TableTop	Insert	Grasp and insert the object	The manipulated object is seated in the target slot.
TableTop	PickPlace	Pick and place the object	The object is released and remains inside the target region.
TableTop	Pour	Grasp and pour into the target	The pouring sequence satisfies the task completion condition.
Mobile	Mobile Reach	Reach a target with base–arm coordination	The end effector reaches the target pose within tolerance.
Mobile	Mobile Grasp	Approach and grasp the target	The target is securely grasped after mobile manipulation.
Mobile	Mobile Open Drawer	Approach and open a drawer	The drawer joint reaches the task-specific open threshold.
Mobile	Mobile Close Drawer	Approach and close a drawer	The drawer joint reaches the task-specific closed threshold.

E.1.3 TASK-SPECIFIC PLAYBENCH INTERVENTIONS

PlayBench CF interventions modify a task-level pose, direction, magnitude, contact choice, or action timing. To state the repeated strength schedules compactly, we define

$$\begin{aligned} P &= (0.01, 0.02, 0.04)\ \mathrm{m}, & A &= (10, 20, 30)\ \mathrm{deg}, \\ T_{\mathrm{close}} &= (-4, -6, -10)\ \mathrm{steps}, & T_{\mathrm{duration}} &= (10, 20, 30)\ \mathrm{steps}, \\ S &= (1.8, 1.6, 1.2), & C &= (2, 4, 6)\ \mathrm{indices}, \end{aligned} \tag{45}$$

where each tuple lists the small, medium, and large settings. Deviations from these shared schedules are written explicitly in Table 11.

The common PlayBench DS pool contains action noise with standard deviations $(0.06, 0.16, 0.30)$, motion noise with standard deviations $(0.04, 0.10, 0.20)$, action-dropout probabilities $(0.10, 0.30, 0.60)$, and control delays of $(2, 4, 8)$ simulation steps. At evaluation, one distur-

bance type and one eligible phase are sampled, and the disturbance remains active for one complete policy action chunk. The last column of Table 11 lists the eligible phases for each task.

Table 11: **Task-specific PlayBench interventions.** CF changes are sampled from the listed task-level variations. DS samples one of the four common execution disturbances and applies it during one listed phase.

Task	Trigger phase	CF variation and strength	Eligible DS phases
Grasp	Grasp	Grasp-position offset P; grasp-yaw offset A; early gripper closing T_{close}	Pre-grasp, grasp, close gripper, retrieval
PushT	Reach, approach, push	Reach-position offset $(0.02, 0.04, 0.08)$ m; reach-yaw offset A; push-direction offset $(30, 50, 70)$ deg; approach-height offset $(0.02, 0.04, 0.08)$ m	Reach, approach, push
RandomWave	Grasp, wave	Grasp variations; wave-position offset P; wave-yaw offset A; wave-duration offset T_{duration}	Grasp phases, wave
Open Drawer	Approach, grasp handle, pull	Pull-direction offset A; pull-distance scale S; handle-contact offset C	Approach, grasp handle, close gripper, pull, release
Close Drawer	Approach, contact, push	Push-direction offset A; push-distance scale S; contact-edge offset C	Approach, push position, close gripper, push
Insert	Grasp, insert	Grasp variations; insertion-position offset P; insertion-yaw offset A; early release $(6, 10, 16)$ steps	Grasp and insertion phases
PickPlace	Grasp, place	Grasp variations; placement-position offset $(0.10, 0.22, 0.36)$ m; placement-yaw offset A; early release $(6, 10, 16)$ steps	Grasp and placement phases
Pour	Grasp, pour, place	Grasp variations; pour-position offset P; pour-yaw offset A; pour-duration offset T_{duration}; placement pose and release-time variations	Grasp, pour, recovery, and placement phases
Mobile Reach	Reach	Target-position offset P; target-yaw offset A	Reach
Mobile Grasp	Grasp	Grasp-position offset P; grasp-yaw offset A; early gripper closing T_{close}	Pre-grasp, grasp, close gripper, retrieval
Mobile Open Drawer	Grasp handle, pull	Pull-distance scale $(1.40, 0.90, 0.50)$; handle-contact offset $(4, 8, 12)$ indices; handle-position offset $(0.20, 0.40, 0.70)$ m	Approach, grasp handle, close gripper, pull, release
Mobile Close Drawer	Contact, push	Push-distance scale S; contact-edge offset C; contact-position offset $(0.06, 0.12, 0.24)$ m	Approach, push position, close gripper, push

E.1.4 RoboTwin-Play Tasks and Interventions

RoboTwin-Play adapts eight bimanual manipulation tasks in SAPIEN and augments them with observable task-level CF changes and end-effector DS perturbations. Table 12 maps the official RoboTwin task names to the concise intervention aliases used in our result and intervention tables. CF is applied after the task reaches the listed critical interaction state. DS perturbs the arm involved in the corresponding interaction; Roller Skid and Box Offset perturb both arms because their corresponding grasp is bimanual. Unlike PlayBench, RoboTwin-Play uses only action noise for Post-DS, applied throughout one complete action chunk at the selected interaction phase.

Table 12: **RoboTwin task-name mapping.** Official names and RoboTwin-Play aliases.

Official task	Alias	Official task	Alias
Click Bell	Bell Miss	Put Bottles Dustbin	Bin Skew
Place Container Plate	Plate Slide	Stack Bowls Three	Bowl Topple
Hanging Mug	Rack Drop	Grab Roller	Roller Skid
Dump Bin Bigbin	Bin Haul	Place Cans Plasticbox	Box Offset

Table 13: **RoboTwin-Play task-specific interventions.** Each adapted task is evaluated under its original success function after the listed CF or DS change.

Task	Adapted objective	Post-CF intervention	Post-DS intervention
Bell Miss	Press a bell	Translate the bell by 0.10 m before contact	Perturb the pressing arm in the horizontal plane with $\sigma = 0.06$
Plate Slide	Place a container onto a plate	Translate the plate by 0.14 m before placement	Perturb the placing arm with $\sigma = 0.06$
Rack Drop	Hang a mug on a rack	Translate the rack by 0.12 m before hanging	Perturb the hanging arm with $\sigma = 0.06$
Bin Haul	Empty a small bin into a large bin	Translate the large bin by 0.30–0.60 m before pouring	Perturb the pouring arm with $\sigma = 0.06$
Bin Skew	Put three bottles into a bin	Translate the bin by 0.16 m before the first drop	Perturb the dropping arm along the horizontal axis with $\sigma = 0.08$
Bowl Topple	Stack three bowls	Translate the base bowl by 0.12 m before stacking the upper bowls	Perturb the final placement with $\sigma = 0.10$
Roller Skid	Grasp and lift a roller with both arms	Translate the roller by 0.12–0.16 m before grasping	Perturb both grasping arms with $\sigma = 0.06$
Box Offset	Place two cans into a plastic box	Translate the box by 0.14 m before placement	Perturb both grasping arms with $\sigma = 0.06$

The RoboTwin-Play training and evaluation disturbances share the same physical semantics but enter through different execution interfaces. During play-data generation, DS modifies the expert execution target. During closed-loop evaluation, DS constructs a perturbed target from the current end-effector state, so that the intervention does not require access to the expert's future actions. The policy observes the resulting state and replans after the perturbed control interval.

E.1.5 Real-World Task Instances

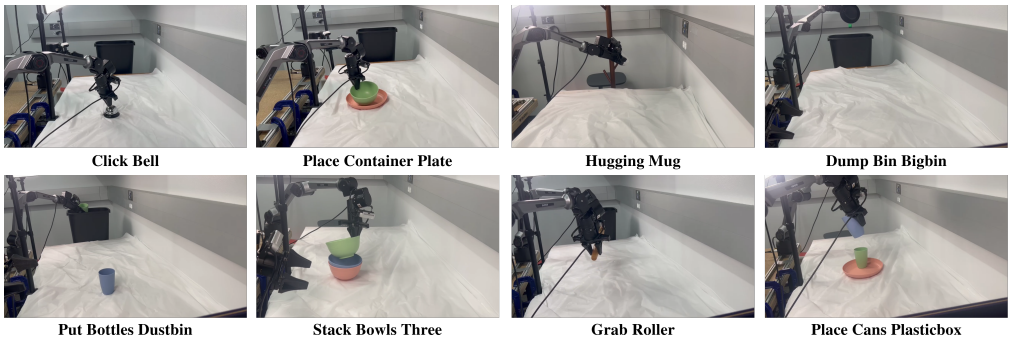


Figure 7: **Real-world task instances.** The eight labeled tasks use the same Piper-based ALOHA platform and correspond to the evaluation in Fig. 6.

E.1.6 LIBERO STANDARD EVALUATION

LIBERO uses its standard Spatial, Object, Goal, and Long suites, with ten tasks per suite. We use the standard demonstrations, language instructions, task initialization, and success evaluators for all 40 tasks. No custom CF or DS intervention and no simulator-generated PlayData are added. LIBERO therefore measures the contribution of future evaluation and candidate selection under a fixed standard-data setting, separately from the intervention robustness tested by PlayBench and RoboTwin-Play.

E.2 PER-TASK SUCCESS RATES

This section expands the aggregate success rates in the main paper into task-level results. We report the same three evaluation conditions defined in Appendix E.1.1. All entries are success rates in percent, computed from 100 evaluation episodes for each task and setting. The TableTop, Mobile, and overall averages in Table 14 correspond directly to the nine model–data combinations in the main PlayBench table; the eight-task averages in Table 15 correspond to the main RoboTwin-Play table.

E.2.1 PLAYBENCH PER-TASK RESULTS

Table 14 reports all twelve PlayBench tasks for the three training sets used by each model. Play-Success is the naturally occurring successful subset of Play-Full. We separate TableTop and Mobile tasks within each evaluation condition so that the group averages can be checked directly against the main-paper results.

Table 14: **Per-task success rates on PlayBench.** Success rate (%) over 100 trials per task and setting. Nom., Succ., and Full denote training on Nominal, Play-Success, and Play-Full, respectively.

	DreamZero-5B			**WAV**			**PlayWAM**		
Task	Nom.	Succ.	Full	Nom.	Succ.	Full	Nom.	Succ.	Full
(a) Nominal evaluation									
TableTop									
Grasp	65	76	52	78	84	47	83	83	85
PushT	61	67	37	72	67	36	73	72	77
RandomWave	52	64	37	62	81	40	66	66	65
Close Drawer	69	82	51	70	77	43	74	88	89
Open Drawer	43	39	21	52	52	25	42	57	50
Insert	44	33	23	32	34	21	47	52	47
PickPlace	47	50	23	52	51	29	48	52	60
Pour	30	29	30	35	30	11	33	42	47
TableTop Avg.	51.4	55.0	34.3	56.6	59.5	31.5	58.3	64.0	65.0
Mobile									
Mobile Reach	80	83	52	78	90	69	88	86	91
Mobile Grasp	69	78	42	76	82	51	59	77	82
Mobile Open Drawer	46	62	36	54	58	34	54	53	70
Mobile Close Drawer	85	74	68	76	87	58	84	86	80
Mobile Avg.	70.0	74.3	49.5	71.0	79.3	53.0	71.3	75.5	80.8
Overall Avg.	57.6	61.4	39.3	61.4	66.1	38.7	62.6	67.8	70.3
(b) Post-CF evaluation									
TableTop									
Grasp	36	32	14	46	50	19	40	51	65
PushT	32	43	16	38	42	7	49	44	69
RandomWave	27	27	10	33	38	17	36	37	53
Close Drawer	36	48	16	36	36	11	43	68	64
Open Drawer	24	19	5	29	23	11	22	32	42
Insert	21	17	8	16	21	7	24	26	39
PickPlace	24	25	9	24	32	6	16	26	48
Pour	13	16	6	14	10	4	14	22	37
TableTop Avg.	26.6	28.4	10.5	29.5	31.5	10.3	30.5	38.3	52.1
Mobile									
Mobile Reach	46	34	16	41	58	26	51	54	75
Mobile Grasp	33	43	12	43	42	18	31	49	74
Mobile Open Drawer	23	25	10	27	25	11	32	32	58
Mobile Close Drawer	36	30	22	35	47	24	44	51	70

Continued on the next page.

Table 14 continued from the previous page.

	DreamZero-5B			WAV			PlayWAM		
Task	Nom.	Succ.	Full	Nom.	Succ.	Full	Nom.	Succ.	Full
Mobile Avg.	34.5	33.0	15.0	36.5	43.0	19.8	39.5	46.5	69.3
Overall Avg.	29.3	29.9	12.0	31.8	35.3	13.4	33.5	41.0	57.8
(c) Post-DS evaluation									
TableTop									
Grasp	19	26	7	32	39	13	31	44	60
PushT	21	30	6	28	28	3	40	31	59
RandomWave	19	18	6	24	28	10	30	29	46
Close Drawer	24	34	7	21	26	4	31	50	55
Open Drawer	17	13	2	19	16	5	17	22	34
Insert	16	11	3	10	15	3	19	20	33
PickPlace	16	18	4	17	23	2	12	20	40
Pour	9	10	2	8	6	1	10	17	29
TableTop Avg.	17.6	20.0	4.6	19.9	22.6	5.1	23.8	29.1	44.5
Mobile									
Mobile Reach	35	25	7	30	45	14	39	43	64
Mobile Grasp	20	33	5	28	29	11	24	40	66
Mobile Open Drawer	12	17	5	17	16	5	25	25	48
Mobile Close Drawer	22	20	10	21	33	12	31	43	61
Mobile Avg.	22.3	23.8	6.8	24.0	30.8	10.5	29.8	37.8	59.8
Overall Avg.	19.2	21.3	5.3	21.3	25.3	6.9	25.8	32.0	49.6

The task-level breakdown measures whether the aggregate behavior is consistent across manipulation skills and embodiments. In particular, the TableTop and Mobile average rows expose any difference between fixed-base manipulation and mobile base–arm coordination, while the three panels separate nominal task competence from robustness to task-level changes and execution disturbances.

E.2.2 RoboTwin-Play Per-Task Results

Table 15 expands the RoboTwin-Play averages for the eight adapted tasks. The Normal–Play-Full comparison is made within each of the three model families: architecture and initialization are held fixed, while the training set changes from nominal demonstrations to nominal plus structured play trajectories.

Table 15: **Per-task success rates on RoboTwin-Play.** Success rate (%) over 100 trials per task and setting under Nominal, Post-CF, and Post-DS evaluation. Normal uses nominal demonstrations only; Play-Full uses nominal demonstrations and the complete structured-play set.

	DreamZero-5B		WAV		PlayWAM	
Task	Normal	Full	Normal	Full	Normal	Full
(a) Nominal						
Bell Miss	84	72	93	73	94	95
Plate Slide	99	85	95	78	99	98
Rack Drop	15	14	19	11	30	37
Bin Haul	78	59	80	62	84	86
Bin Skew	47	40	52	44	50	54
Bowl Topple	76	67	77	62	81	81
Roller Skid	86	72	89	75	93	94
Box Offset	86	74	91	71	94	96
Average	71.4	60.4	74.5	59.5	78.1	80.1
(b) Post-CF						
Bell Miss	38	25	40	21	45	75
Plate Slide	44	18	48	29	54	83
Rack Drop	1	0	8	0	17	31
Bin Haul	31	16	33	17	29	66
Bin Skew	23	3	26	9	33	47
Bowl Topple	25	7	29	12	35	73
Roller Skid	30	13	34	17	47	77
Box Offset	33	25	41	24	44	76
Average	28.1	13.4	32.4	16.1	38.0	66.0

Continued on the next page.

Table 15 continued from the previous page.

	DreamZero-5B		WAV		PlayWAM	
Task	Normal	Full	Normal	Full	Normal	Full
(c) Post-DS						
Bell Miss	36	13	34	13	38	68
Plate Slide	36	17	39	19	46	79
Rack Drop	1	0	4	0	11	26
Bin Haul	25	11	27	11	24	69
Bin Skew	23	3	19	0	27	46
Bowl Topple	23	7	23	8	29	71
Roller Skid	25	12	30	10	33	79
Box Offset	33	22	35	17	41	72
Average	25.3	10.6	26.4	9.8	31.1	63.8

The per-task results support within-family comparisons between Full and Normal for DreamZero-5B, WAV, and PlayWAM. These measure the contribution of the complete structured-play set under each model's training objective. Comparing the three Full variants then shows how the models use the same structured trajectories. Each panel average is computed over all eight tasks and matches the corresponding aggregate in the main paper.

E.3 Training Data, Settings, and Evaluation Domains

The three benchmarks play different experimental roles and therefore use different data controls. PlayBench compares Nominal, Play-Success, and Play-Full training. Nominal and Play-Full have exactly matched trajectory counts, while Play-Success is the successful subset naturally present in Play-Full. RoboTwin-Play likewise compares count-matched Normal and Play-Full sets in a second simulator. LIBERO holds the standard training data fixed and follows the official protocol. Table 16 summarizes these settings.

Table 16: **Training-data composition and evaluation domains.** Counts are given per task and over the tasks used by each benchmark. A PlayBench branch group contains one trajectory from each of the five PlayData modes.

Benchmark	Variant	Training data per task	Traj./task	Total traj.	Evaluation domains
PlayBench	Nominal	1,000 successful Nominal trajectories	1,000	12,000	Nominal, Post-CF, Post-DS
PlayBench	Play-Success	Successful subset of the same Play-Full groups	–	–	Nominal, Post-CF, Post-DS
PlayBench	Play-Full	200 complete branch groups $\times$ 5 modes	1,000	12,000	Nominal, Post-CF, Post-DS
RoboTwin-Play	Normal	150 official Nominal trajectories	150	1,200	Nominal, Post-CF, Post-DS
RoboTwin-Play	Play-Full	50 Nominal + 25 CF pairs + 25 DS pairs	150	1,200	Nominal, Post-CF, Post-DS
LIBERO	Standard	Official demonstrations	–	–	Official standard evaluation

E.3.1 PlayBench Training and Evaluation Setting

For each of the twelve PlayBench tasks, the collection pool contains 1,000 complete branch groups. We construct three training sets from this pool. The Nominal set takes the successful Nominal trajectory from each of the 1,000 groups, resulting in 1,000 trajectories per task. Play-Full takes a fixed subset of 200 complete branch groups and retains all five trajectories in each group:

$$200 \text{ branch groups} \times 5 \text{ trajectories per group} = 1{,}000 \text{ trajectories per task.} \tag{46}$$

Thus, Nominal and Play-Full each contain 12,000 trajectories over the twelve tasks. Play-Success retains only the outcome-successful trajectories from the same 200 Play-Full groups; it is therefore the naturally occurring successful subset of Play-Full. This comparison separates the benefit of diverse successful play from the additional effect of failure-containing experience. For every data setting, DreamZero-5B, WAV, and PlayWAM receive identical trajectories. The five trajectories of each Play-Full group remain together throughout preprocessing and training.

The 32-task collection described in Appendix B.1.4 is the complete PlayData inventory. The PlayBench experiments use the data from the fixed twelve-task benchmark subset. Training and evaluation seeds are disjoint. Evaluation uses held-out initial robot configurations, object poses, appearances, camera configurations, and other task-specific scene factors. The same evaluation seeds are

shared across methods. Nominal uses the held-out scene without a test-time intervention; Post-CF applies one task-specific state or task-level change; and Post-DS applies one execution disturbance as defined in Appendix E.1.3.

E.3.2 RoboTwin-Play Training and Evaluation Setting

All RoboTwin-Play trajectories are collected in the official clean environment, and evaluation is conducted through the XPolicyLab interface. For each of the eight tasks, Normal contains 150 official Nominal trajectories. Play-Full contains 50 Nominal trajectories and 25 matched CF failure–recovery pairs and 25 matched DS failure–recovery pairs. Each pair contributes a failed continuation and a successful recovery from the matched off-nominal state. The Play-Full set therefore contains

$$50 \text{ Nominal} + (25 \text{ CF pairs}) \times 2 + (25 \text{ DS pairs}) \times 2 = 150 \text{ trajectories per task.} \tag{47}$$

Across eight tasks, both Normal and Play-Full contain 1,200 trajectories; Play-Full includes 400 successful recoveries and 400 failed continuations. Because the CF and DS data are constructed as matched failure–recovery pairs, the pair remains intact and we do not form a separate Play-Success condition. Within each WAV or PlayWAM comparison, the architecture, initialization, trajectory count, and evaluation protocol are held fixed; only data composition changes.

Nominal evaluation uses the official clean test environment. Post-CF and Post-DS begin from the same clean environment and introduce the task-specific interventions listed in Appendix E.1.4. The same held-out initializations and intervention specifications are used for all compared variants. Training-time and evaluation-time DS share the same physical meaning while using the execution interfaces described in Appendix E.1.4.

E.3.3 LIBERO Standard Setting

For LIBERO, we follow the official training and evaluation setting for the Spatial, Object, Goal, and Long suites. All methods use the official demonstrations, language instructions, task initializations, and success evaluators. We do not add PlayData, CF/DS branches, or custom test-time interventions. The compared PlayWAM variants use the same standard training data, so this benchmark isolates the contribution of value-aware candidate selection under a fixed-data setting.

E.4 Additional Ablations and Analyses

The main PlayBench experiment already compares Nominal, Play-Success, and Play-Full for each model family. The following experiments start from PlayWAM trained on Play-Full and separate the contributions of branch-group structure, branch-specific supervision, recovery experience, and value-based action selection. Unless stated otherwise, every variant uses the same backbone, initialization, trajectory count, optimization schedule, and inference configuration as the full model.

E.4.1 PlayData Structure and Branch-Specific Supervision

Table 17 tests which parts of the PlayData construction are responsible for the gain. *All-action imitation* applies the action objective to every recorded trajectory, including actions that lead to failure, instead of masking action supervision on failed branches.

The next two variants preserve data volume. *Shuffled branch groups* randomly permutes continuations across anchor states within each task before constructing the value comparisons. We never pair branches across tasks, which would introduce trivial differences in task difficulty and utility scale. The trajectories and supervision masks remain unchanged, but the value objective no longer receives matched same-state comparisons. *Recovery → Nominal* replaces each successful Recovery branch with an additional Nominal branch, preserving the number of successful trajectories while removing explicit retry behavior. The full model uses all branch types for dynamics learning, successful executable segments for action learning, and outcome comparisons within each branch group for value learning.

Table 17: **Ablation of PlayData structure and supervision.** All result columns report success rate (%). "Same group" denotes value comparisons between continuations collected from a shared anchor state.

Variant	Dynamics data	Action targets	Value relation	Nom.	Post-CF	Post-DS	Robust Avg.
All-action imitation	All branches	All branches	Same group	58.1	25.9	18.9	22.4
Shuffled branch groups	All branches	Successful executable segments	Shuffled	62.4	51.3	44.0	47.7
Recovery → Nominal	Count-matched branches	Successful executable segments	Same group	65.4	53.7	46.1	49.9
Full PlayWAM	**All branches**	**Successful executable segments**	**Same group**	**70.3**	**57.8**	**49.6**	**53.7**

Together, these controls distinguish three possible explanations for the improvement: exposure to diverse state transitions, direct imitation of every recorded action, and the matched failure–recovery structure of a branch group. We report the nominal score separately from the two post-intervention scores so that robustness gains cannot hide a loss of standard task competence. Robust Avg. is the unweighted mean of Post-CF and Post-DS success.

E.4.2 Value Learning and Candidate Selection

Table 18 compares random, action-likelihood, learned-value, and oracle selection on identical candidate sets from the same model. The oracle executes every candidate from the cloned simulator state and selects the one with the highest ground-truth utility; it is an evaluation-only upper bound.

Table 18: **Candidate-selection analysis.** PlayBench uses $B = 4$; LIBERO reports $B \in \{4, 8\}$. All entries are success rates (%).

Selection rule	PB Nom.	PB Post-CF	PB Post-DS	LIBERO $B = 4$	LIBERO $B = 8$
Random	63.5	49.6	43.8	98.1	98.2
Action likelihood	66.0	53.2	47.5	98.3	–
PlayWAM value	70.3	57.8	49.6	98.9	99.0
Oracle	**71.5**	**58.6**	**51.2**	**99.0**	**99.1**

Random selection gives the Play-Full generator a 46.7% Robust Avg., above Play-Success's 36.5%, showing that Play-Full improves generation before ranking. Learned value selection further raises robustness to 53.7%, close to the oracle's 54.9%. On LIBERO, the value selector is within 0.1 point of the oracle for both budgets, while the oracle itself improves only from 99.0% to 99.1% at $B = 8$. The limited additional headroom and lower inference rate support $B = 4$ as the default.

E.4.3 Held-Out Disturbance Generalization

We test whether robustness transfers to a disturbance family that is absent from training. For each row of Table 19, we exclude one DS type, resample the remaining DS types to keep the number of training trajectories fixed, and evaluate the excluded type at medium strength. The in-domain score uses the corresponding full-data model under the same test condition; the held-out score uses the leave-one-type-out model.

Table 19: **Generalization to unseen execution disturbances on PlayBench.** Drop is the absolute success-rate difference between the in-domain and held-out training conditions.

Held-out DS type	In-domain Post-DS	Held-out Post-DS	Drop
Action noise	51.6	41.2	10.4
Motion noise	47.9	32.5	15.4
Action dropout	58.3	49.4	8.9
Control delay	41.0	28.6	12.4