DREAMEVOLVE: GENERALIZABLE ROBOT POLICY SELF-EVOLUTION THROUGH LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalist robot policies have shown impressive manipulation skills, but deploying them reliably in new scenes and tasks remains difficult. Real-world evaluation and improvement require extensive rollouts and human demonstrations, both of which are slow and difficult to scale. World models offer a scalable alternative by allowing policies to roll out in imagination, yet existing pixel-level models entangle task dynamics with appearance and struggle to provide reliable training signals across diverse scenes and novel tasks. We present **DreamEvolve**, a robotic self-evolution framework built on a controllable latent world model. A Joint-Attention-of-Mixed-Experts architecture learns action-conditioned latent dynamics, while Latent Progress Reward Optimization efficiently converts imagined rollout groups into group-relative progress signals. Across simulation and real-world experiments, DreamEvolve substantially improves policy robustness under out-of-distribution shifts, including cluttered scenes, long-horizon compositional tasks, and challenging deployment conditions. On LIBERO-Pro, DreamEvolve achieves strong cross-scene, cross-task, and out-of-distribution generalization after a single round of imagination-based post-training. DreamEvolve also raises the average success rate under real-world perturbations from 10.1% to 61.7%. We also open-source **DreamEvolveBench**, the first large-scale paired scene-variation dataset for controllable robotic world models.

1 INTRODUCTION

Recent vision-language-action (VLA) models have established broad manipulation capabilities across tasks, scenes, and robot embodiments (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025; Team et al., 2024). Open-world deployment remains challenging because policy execution in the physical world is often brittle. Existing approaches typically address such failures by collecting additional expert trajectories, but even policies trained with these human demonstrations can still remain unreliable when deployment conditions change (Shi et al., 2025; Intelligence et al., 2025). Real-world physical interaction is also expensive and effectively irreversible: each rollout consumes hardware time and commits to one realized trajectory. Together, these constraints make it costly to expose and correct brittle behavior through repeated physical trials (Atreya et al., 2025; Zhou et al., 2025b). Using a world model for robotic self-evolution offers a different path: it generates large-scale, low-cost policy rollouts and turns imagined interactions into direct feedback for policy improvement.

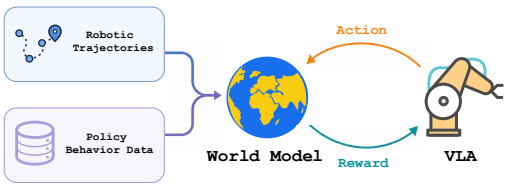


Figure 1: **DreamEvolve Overview.** Robot trajectories and target-robot data train the world model at scale.

However, existing world-model approaches for policy self-evolution predominantly operate in pixel space (Yang et al., 2026; Jang et al., 2025; Zhu et al., 2026; 2025; Guo et al., 2026). Pixel prediction couples action consequences with texture, lighting, background, and camera detail, forcing the model to learn substantial visual knowledge unrelated to action execution. These nuisance attributes can also perturb the predicted trajectory even when the underlying interaction dynamics remain unchanged. A latent world model offers a more direct path by predicting action-conditioned futures in the visual

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.

representation on which the policy already acts. It can vary the imagined world while retaining control-relevant semantics and dynamics, then return each predicted state directly to the policy. This policy-native interface supports scalable imagination for transferable behavior.

This raises a central question: **Can robotic self-evolution be achieved directly through latent-space imagination, and can such imagination provide more reliable and generalizable representations than pixel-space rollouts?** To answer this question, we introduce **DreamEvolve**, an end-to-end framework that turns latent imagination into a closed-loop engine for policy improvement, as illustrated in Figure 1. The world model jointly understands language instructions, supports text- and reference-guided scene variation, and preserves robot-object interaction dynamics in latent space. A staged training recipe grounds these capabilities in action-free videos, egocentric interaction videos, heterogeneous robot trajectories, and target-robot data. After a single round of policy improvement, DreamEvolve delivers robust execution across changes in background, object position, lighting, and other visual or spatial conditions, together with strong long-horizon compositional performance in simulation and the real world.

Our experiments establish robust long-horizon compositional execution and strong out-of-distribution generalization through imagination-based post-training. Our main contributions are fourfold: (i) we achieve strong cross-task, cross-scene, and out-of-distribution generalization in both simulation and the real world; (ii) we design a **Joint-Attention-of-Mixed-Experts** architecture that supports self-supervised action-conditioned latent dynamics learning, enabling generated latent representations to reflect the causal effects of actions on future states; (iii) we introduce **Latent Progress Reward Optimization (LPRO)**, a reinforcement learning algorithm for latent-space policy improvement that efficiently converts imagined rollout groups into dense learning signals; and (iv) we open-source **DreamEvolveBench**, the first large-scale paired scene-variation dataset for controllable robotic world models, containing one million generated video trajectory variants with structured changes in scene attributes.

2 Related Work

Generalist Robot Manipulation Policies. A line of work has converged on training a single policy that controls many robots, in many scenes, from natural-language instructions, by scaling imitation learning over heterogeneous teleoperation data and, more recently, by initializing from pretrained vision-language backbones to inject web-scale semantic knowledge into control (Brohan et al., 2022; Chi et al., 2025; Team et al., 2024; O'Neill et al., 2024; Khazatsky et al., 2024; Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025; Shi et al., 2025). Despite striking gains, two limitations are now widely acknowledged: meaningfully evaluating such policies requires hundreds of physical rollouts whose cost grows faster than the policies themselves (Atreya et al., 2025; Zhou et al., 2025b), and the dominant remedy for failure cases remains collecting more expert teleoperation, which scales poorly with the long tail of real-world objects, scenes, and instructions (Shi et al., 2025; Intelligence et al., 2025). DreamEvolve targets this gap by letting a generalist policy practice and improve on its own, inside a controllable internal model of the worlds it must learn to operate in.

World Models for Policy Learning. Improving policies inside learned world models is rooted in latent-dynamics RL (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2023; Wu et al., 2023). Recent manipulation methods use action-conditioned video world models for policy evaluation, synthetic-data generation, planning, and policy optimization (Zhu et al., 2025; Guo et al., 2026; Jang et al., 2025; Yang et al., 2026; Zhu et al., 2026). Complementary work predicts future features for representation learning, planning, or policy refinement (Assran et al., 2025; Sun et al., 2026; Huang et al., 2025; Chandra et al., 2025; Jiang et al., 2025). DreamEvolve connects these directions through a policy-facing latent dynamics model with text- and reference-controlled interventions and closed-loop policy optimization.

3 Data Generation and Benchmark

To train a world model that allows a policy to rehearse the *same* action under *multiple variations of the same world*, the data must pair identical action sequences with systematic scene changes. However,

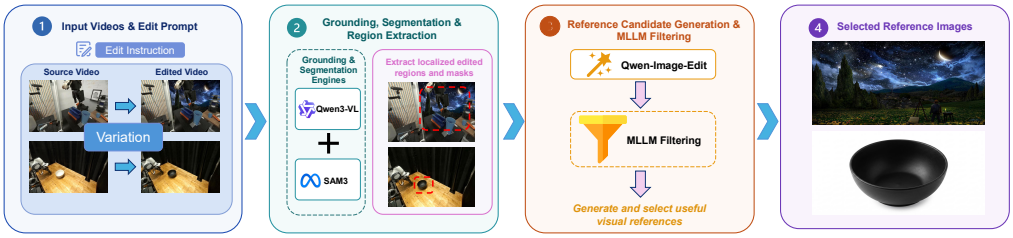


Figure 2: **Reference-Image Generation Pipeline.** Given a source video and edit instruction, we first synthesize a scene variation and localize edited regions with grounding and segmentation models. We then generate reference-image candidates with an image editor and use MLLM filtering to retain instruction-aligned reference images.

existing manipulation datasets do not provide this structure. Real in-the-wild robot benchmarks (Khazatsky et al., 2024; Walke et al., 2023; O'Neill et al., 2024) contain diverse scenes, but each scene usually involves different tasks, objects, and trajectories, offering no paired supervision for same trajectory, different background or object/attribute. Simulation and generalization benchmarks (Nasiriany et al., 2024; Chen et al., 2025; Morgan et al., 2026) provide domain-randomized variants, but these are primarily in simulation. We therefore introduce **DreamEvolveBench**, the first large-scale paired scene-variation dataset for controllable robotic world models, containing 1M trajectories across diverse tasks. Each base trajectory is paired with structured variants that preserve the same action sequence while editing the background, manipulated and surrounding objects, lighting, table material, and viewpoint; each variant is also annotated with a natural-language edit instruction and a reference image. We use pretrained Editto (Bai et al., 2026) to generate edited videos and the Kiwi-Edit reference-image pipeline (Lin et al., 2026) to construct reference images. Our main contribution is the construction of a large-scale paired scene-variation dataset built from carefully filtered real-robot trajectories from Open X-Embodiment (O'Neill et al., 2024).

Scalable Data Generation Pipeline. For each real-robot trajectory, we form a source tuple comprising source video V_s, recorded action sequence $\mathbf{a}_{1:T}$, and task instruction l.

We generate a dense caption c, derive edit instruction l_{edit}, and use pretrained Editto to synthesize edited video V_e from V_s and l_{edit}. The edited video varies background, lighting, surrounding objects, or viewpoint while preserving robot motion. We then generate reference image r from V_e using l_{edit}. For global edits, r is a robot-removed background reference; for local object edits, it is an object-centric reference from the grounded and masked edited object. The final sample is $(V_s, V_e, \mathbf{a}_{1:T}, l, c, l_{\text{edit}}, r)$, where r is the reference image aligned with the edit.

Quality Control and Post-Processing. We apply the same final quality-control stage to ensure that each tuple is suitable for world-model training. Gemini-3.1-Pro (Google DeepMind, 2026) serves as the VLM judge, filtering samples by instruction fidelity, visual quality, temporal consistency, reference alignment, and manipulation preservation. We discard low-quality or inconsistent variants that would compromise world-model training. We further deduplicate reference images using CLIP (Radford et al., 2021) features to reduce redundancy and prevent data leakage. For each source video, we retain 5–30 variants spanning diverse edit attributes. We ultimately obtain 1M generated video trajectory variants spanning heterogeneous data from diverse robot embodiments.

4 Methodology

4.1 Model Architecture and Challenges

Challenge 1: Controllable Latent Dynamics. A policy-facing latent dynamics model must jointly support reference-guided latent editing and action-conditioned future prediction from task and edit instructions, reference images, and other visual conditions, together with robot actions. Existing robot world models commonly predict pixel trajectories (Zhu et al., 2025; Guo et al., 2026; Zhu et al., 2026), while DreamGen generates videos that require separate action recovery (Jang et al., 2025). Neither route provides a single policy-facing latent model for reference-guided latent editing and action-

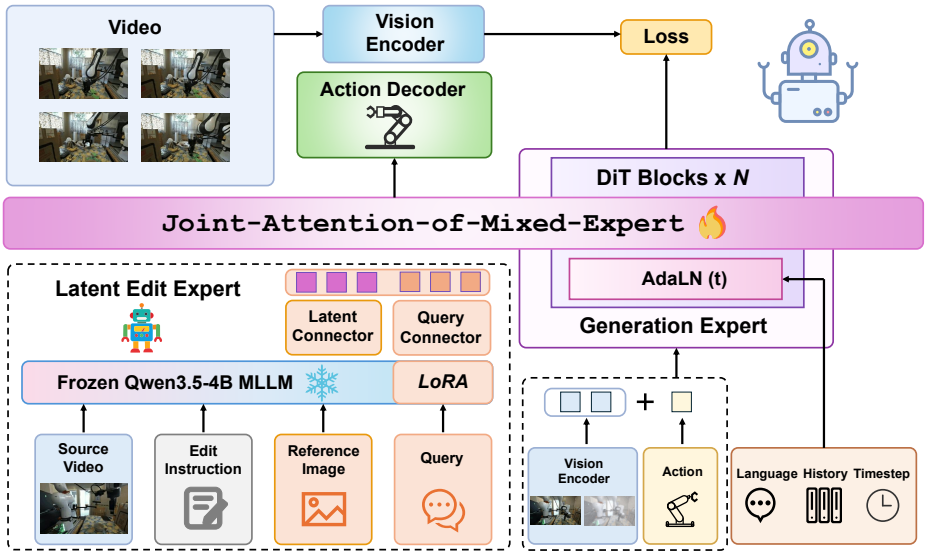


Figure 3: **DreamEvolve Architecture.** DreamEvolve couples a pretrained VLM, action decoding, and a video generation expert through the Joint-Attention-of-Mixed-Experts architecture. LPRO improves the policy from closed-loop imagined rollouts.

conditioned future prediction. Consequently, jointly modeling these capabilities while grounding the model in action-free videos, egocentric interaction videos, heterogeneous robot trajectories, and target-robot data remains a central challenge, which is precisely the gap our work aims to fill.
Challenge 2: Efficient Policy Improvement. Another central challenge is how to make policy improvement efficiently benefit from partial progress at every stage of an imagined rollout. LPRO addresses this challenge by converting instruction-conditioned latent progress into dense action-level learning signals. With these goals, DreamEvolve operates in the policy visual latent space. Let o_t denote the RGB observation at time t, $V_{\le t} = (o_0, \dots, o_t)$ the observed video context, and E_π the frozen vision encoder of policy π, mapping each observation to policy visual latent $h_t = E_\pi(o_t)$. We denote the action-conditioned latent world model, with parameters ϕ, by W_ϕ.

Joint-Attention-of-Mixed-Experts. Our architecture contains two Transformer experts: a *Latent-Edit Expert* and a *Generation Expert*. The *Latent-Edit Expert* encodes the observed video context $V_{\le t}$, reference image r, and edit instruction l_{edit} into scene-variation tokens X_{edit}. The *Generation Expert* is initialized from a Wan2.2-TI2V-5B (Wan et al., 2025) diffusion-transformer backbone and takes policy visual latents, the task instruction l, action history, a masked action chunk, and a flow timestep as input. At each layer, let Q_z, K_z, and U_z denote the query, key, and value matrices produced by expert $z \in \{\text{edit}, \text{dyn}\}$, respectively. Each expert keeps its own query while attending to the concatenated keys and values from both experts, giving $\text{JA}_z = \text{Attn}\big(Q_z, [K_{\text{edit}}; K_{\text{dyn}}], [U_{\text{edit}}; U_{\text{dyn}}]\big)$, where Attn is the standard attention operation, $[\,;\,]$ denotes token-sequence concatenation, and JA_z is the joint-attention output for expert z. The attended features are then passed back to their corresponding expert-specific FFN blocks. This allows reference-guided scene-editing tokens to influence latent generation through shared attention, while the generation pathway remains action-conditioned and predicts future latent dynamics.

Generation Expert. The Generation Expert is trained as a conditional-flow model in the policy feature space. For a prediction horizon of K future steps, the frozen encoder E_π, applied framewise, provides the target latent sequence $h^*_{t+1:t+K} = E_\pi(o_{t+1:t+K})$, where * marks a target. For $s \sim \mathcal{U}[0,1]$ and $\xi \sim \mathcal{N}(0, I)$, we form the flow state $h^{(s)}_{t+1:t+K} = (1-s)\xi + s h^*_{t+1:t+K}$, whose target velocity is $h^*_{t+1:t+K} - \xi$. The observed visual latents, $h^{(s)}_{t+1:t+K}$, and an action chunk are encoded as tokens, projected to the Wan2.2-TI2V-5B hidden dimension, and concatenated before entering the DiT backbone. The task instruction l and edit instruction l_{edit} provide semantic conditioning, while s

is injected through AdaLN. A latent output head predicts the conditional velocity v_ϕ in the SigLIP visual-token space, optimized with $\mathcal{L}_{\text{roll}} = \mathbb{E}_{s,\xi}\left[\left\| v_\phi\left(h^{(s)}_{t+1:t+K}, s \mid \cdot\right) - \left(h^{*}_{t+1:t+K} - \xi\right)\right\|_2^2\right]$. At inference, integrating this velocity field from a Gaussian initial state yields the future latent sequence $\hat{h}_{t+1:t+K}$.

Latent-Edit Expert. The Latent-Edit Expert uses a frozen Qwen3.5-4B(Qwen Team, 2026) backbone with LoRA adapters to encode the observed video context $V_{\leq t}$, reference image r, and edit instruction l_{edit}. We append learnable query tokens to the input sequence comprising $V_{\leq t}$, r, and l_{edit}, then map their final MLLM hidden states to the Wan2.2-TI2V-5B hidden dimension with the Query Connector, a two-layer MLP with a zero-initialized final layer. The final MLLM hidden states of the reference visual tokens are mapped by an independent Latent Connector of the same form. The two outputs are directly concatenated as X_{edit}, following the dual-connector conditioning procedure of Kiwi-Edit (Lin et al., 2026). Denoting this learned edit encoder by f_{edit}, its output is $X_{\text{edit}} = f_{\text{edit}}(V_{\leq t}, r, l_{\text{edit}})$. When r or l_{edit} is unavailable in early-stage training, we replace it with a learnable null-condition embedding and keep the same interface. The produced X_{edit} serves as the latent-edit context that interacts with the Generation Expert through cross-expert joint attention.

4.2 Model Training and Data

We design a four-stage pyramid training recipe for DreamEvolve, as illustrated in Figure 4, where each stage corresponds to one level of the data pyramid. Stage 1 trains on action-free web videos, Stage 2 trains on egocentric videos to strengthen interaction priors, Stage 3 trains on heterogeneous robot trajectories with actions, and Stage 4 adapts the model to target-robot data for embodiment-specific specialization.

Stage 1: Learning Latent Visual Dynamics. We first train DreamEvolve on large-scale action-free web videos collected from platforms such as YouTube, TikTok, X, and so on. These videos do not contain robot action labels, so we replace the action input with a learnable null-action token and train the model to predict future visual latents from observed visual context alone. This stage adapts the Generation Expert to learn broad visual dynamics priors, enabling it to predict plausible future visual latents from observed visual context before introducing robot-specific action supervision.

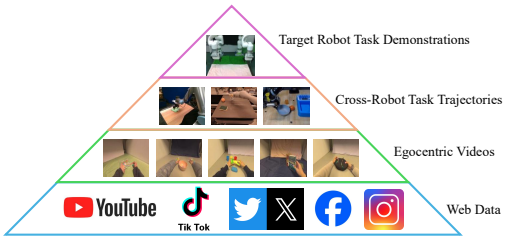


Figure 4: **Four-Stage Data Pyramid.** DreamEvolve progresses from web data and egocentric videos to heterogeneous robot trajectories and target-robot data.

Stage 2: Learning from Egocentric Videos. We then train DreamEvolve on large-scale egocentric videos. These videos are still action-free, so we again use a learnable null-action token and train the model to predict future visual latents from observed visual context. Egocentric videos will show how people interact with objects from a close first-person view.

Stage 3: Learning from Heterogeneous Robot Data. We then train DreamEvolve on our generated robot trajectory variants with real robot actions. Since different robots use different action spaces, we convert their actions into a normalized delta end-effector format. During training, the model predicts future visual latents and also reconstructs masked actions. We mask continuous action segments or selected action dimensions, so the model cannot rely on simple interpolation. This stage teaches the Generation Expert how robot actions affect future states, while the Latent-Edit Expert supplies the scene and reference conditions. Let $\mathcal{L}$ denote the total Stage 3 objective function, let $\mathcal{L}_{\text{act}}$ denote masked-action reconstruction, and let λ_{roll} and λ_{act} be their loss weights. The Stage 3 objective function is $\mathcal{L} = \lambda_{\text{roll}}\mathcal{L}_{\text{roll}} + \lambda_{\text{act}}\mathcal{L}_{\text{act}}$.

Stage 4: Target-Robot Finetuning. Finally, we finetune DreamEvolve on the DROID (Khazatsky et al., 2024) subset of our generated trajectory variants, which matches our Franka target robot

setting (Haddadin et al., 2022). In this stage, we feed the full action sequence as input and no longer reconstruct masked actions.

4.3 On-Policy Reinforcement Learning with Latent Progress Reward Optimization

Reinforcement learning in a latent world model faces two key challenges. **(1) Latent-Space Reward Estimation.** Existing robotic rewards typically rely on visual observations or terminal outcomes, whereas DreamEvolve rolls out directly in policy visual latent space. Decoding imagined states back to pixels only for reward evaluation would sacrifice this efficiency. **(2) Temporal Credit Assignment.** Trajectory-level success rewards cannot distinguish partial progress or identify which actions drive it, limiting standard group-based policy optimization on long-horizon tasks.

To address the challenges above, we introduce **Latent Progress Reward Optimization (LPRO)**, an on-policy RL method that distills instruction-conditioned progress into latent space and converts progress increments into time-aligned group-relative advantages. After training the latent world model, we improve the policy through closed-loop imagined rollouts without collecting new real-world demonstrations. Let π_θ^{pv} denote the policy after its frozen vision encoder, with trainable parameters θ; it maps policy visual latents and task instruction l to actions.

Latent Progress Reward. Following the token-probability formulation of TOPReward (Chen et al., 2026), we train a teacher reward model R_ψ, where ψ denotes its parameters. It estimates instruction-conditioned robotic task progress from video-VLM token probabilities and achieves a mean Value-Order Correlation of 0.958. For each task instruction l, reward distillation uses a group of G recorded trajectories spanning successful, partially successful, and failed behavior. Trajectory i contains a video $V_i = (o_{i,0}, \ldots, o_{i,T})$ and its recorded action sequence $\mathbf{a}_i = (a_{i,1}, \ldots, a_{i,T})$, where i indexes trajectories and T is the recorded-trajectory horizon. At time t, the teacher and latent progress model P_η, with parameters η, score causal prefixes as $R_{i,t}^T = R_\psi(V_{i,0:t}, l)$ and $R_{i,t}^L = P_\eta(h_{i,0:t}, l)$, respectively, where $h_{i,0:T} = E_\pi(V_i)$. For every recorded trajectory, we construct its paired imagined reconstruction from the same initial observation and recorded actions under a null edit context $X_\varnothing$, whose token shape matches X_{edit}. Denote the resulting latent trajectory by $\hat{h}_{i,0:T} = W_\phi(E_\pi(o_{i,0}), \mathbf{a}_i, X_\varnothing, l)$ and its imagined-latent progress score by $R_{i,t}^I = P_\eta(\hat{h}_{i,0:t}, l)$, where superscript I denotes imagination. Let $\tilde{R}_{i,t}^T$, $\tilde{R}_{i,t}^L$, and $\tilde{R}_{i,t}^I$ be the group-normalized progress scores obtained at time t by centering and scaling $R_{i,t}^T$, $R_{i,t}^L$, and $R_{i,t}^I$, respectively, within the recorded-trajectory group, and let λ_{imag} weight supervision on imagined latents. Let $\mathcal{L}_{\mathrm{distill}}$ denote the reward-distillation objective function. We optimize:

$$\mathcal{L}_{\mathrm{distill}} = \frac{1}{GT} \sum_{i=1}^{G} \sum_{t=1}^{T} \Big(\Big\| \tilde{R}_{i,t}^L - \tilde{R}_{i,t}^T \Big\|_2^2 + \lambda_{\mathrm{imag}} \Big\| \tilde{R}_{i,t}^I - \tilde{R}_{i,t}^T \Big\|_2^2 \Big).$$

Progress-score-aligned distillation transfers the teacher's relative progress ordering to both encoded and world-model-predicted latent trajectories at every time step, so P_η scores closed-loop imagined rollouts directly.

Policy Improvement in the World Model. We then use these progress scores to improve the policy through closed-loop latent imagination. Conditioned on a task instruction l, initial policy latent h_0, and a fixed edit context X_{cond}, the current policy samples a group of G closed-loop imagined rollouts $\{\hat{\tau}_i\}_{i=1}^G$ of H action steps, where $X_{\mathrm{cond}} = X_{\mathrm{edit}}$ for an edited state and $X_{\mathrm{cond}} = X_\varnothing$ otherwise. Rollout i is $\hat{\tau}_i = (\hat{h}_{i,0:H}, \hat{\mathbf{a}}_{i,1:H})$, where $\hat{h}_{i,0} = h_0$, $\hat{h}_{i,0:H}$ are its predicted latent states, and $\hat{\mathbf{a}}_{i,1:H}$ are its policy-sampled actions. Each sampled action is returned with X_{cond} and l to W_ϕ, and each predicted latent is returned to π_θ^{pv}. At action step t, the frozen latent progress model produces $p_{i,t} = P_\eta(\hat{h}_{i,0:t}, l)$. The initial progress score is $p_{i,0} = P_\eta(\hat{h}_{i,0}, l)$. We define its stopped-gradient increment as $d_{i,t} = \mathrm{sg}(p_{i,t} - p_{i,t-1})$, where sg stops gradients through the reward and $t = 1, \ldots, H$. With discount factor $\gamma \in [0, 1]$, the progress-to-go return for action $\hat{a}_{i,t}$ is $g_{i,t} = \sum_{k=t}^{H} \gamma^{k-t} d_{i,k}$. For each time step, μ_t and σ_t are the mean and standard deviation of $\{g_{j,t}\}_{j=1}^G$ across the rollout group, and ϵ is a numerical stabilizer. The time-aligned progress advantage is $\hat{A}_{i,t}^{\Delta} = \frac{g_{i,t} - \mu_t}{\sigma_t + \epsilon}$. Thus, each rollout supplies a time-localized learning signal instead of one trajectory-level score. We use

the asymmetric clipped group update from DAPO (Yu et al., 2025) with $\hat{A}^{\Delta}_{i,t}$. Let $\rho_{i,t}(\theta)$ be the conditional-flow surrogate ratio of Flow Policy Optimization (FPO) (McAllister et al., 2026), ϵ_{low} and ϵ_{high} be the lower and upper clipping bounds, and clip bound its first argument to this interval. The expectation is over sampled rollout groups. The LPRO policy objective function, denoted $\mathcal{J}_{\text{LPRO}}(\theta)$, is:

$$\mathcal{J}_{\text{LPRO}}(\theta) = \mathbb{E}\left[\frac{1}{GH}\sum_{i=1}^{G}\sum_{t=1}^{H}\min\Big(\rho_{i,t}(\theta)\hat{A}^{\Delta}_{i,t}, \text{clip}(\rho_{i,t}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\,\hat{A}^{\Delta}_{i,t}\Big)\right].$$

We filter groups with negligible progress variance and update on groups that express a meaningful relative ordering.

5 Experiments

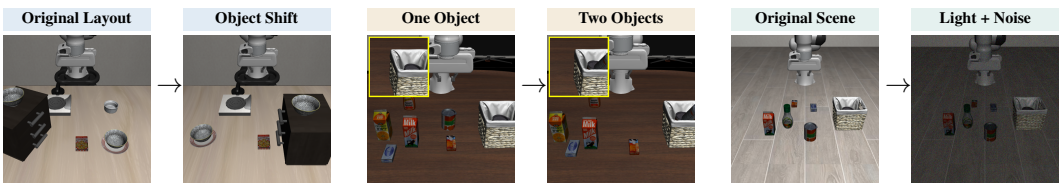
We conduct extensive experiments in both simulation and the real world to evaluate DreamEvolve and answer the following questions. **Q1**: How does DreamEvolve perform in simulation, especially under cross-task, cross-scene, and out-of-distribution generalization? **Q2**: Can existing RL algorithms match the rollout efficiency of DreamEvolve in latent-world-model-based policy self-evolution? **Q3**: How does DreamEvolve perform on real-world robotic manipulation tasks? **Q4**: Can our latent world model produce imagined rollouts whose predicted outcomes reflect real-world task success? **Q5**: Does scene-variant generation improve policy self-evolution and downstream task performance?

Experimental Settings. We train DreamEvolve with the four-stage recipe described in Figure 4; the world model uses eight context and eight target frames. Policy optimization uses $G = 5$ and a 32-step imagination horizon. At post-training, we apply the eight-target-frame predictor autoregressively in closed loop, feeding each predicted latent and newly sampled action back as context to produce the 32-step imagined rollout used by LPRO. All methods use the same SFT initialization, rollout budget, and one post-training round. Simulation results aggregate 100 episodes per condition, and each Franka entry aggregates 100 physical rollouts. For the ablation study, we post-train both π_0 and $\pi_{0.5}$ with $P = 1000$ imagined rollouts per task and report their macro-average across the two policy backbones.

5.1 Simulation Task Experiments

Table 1: **Success Rate (%) on LIBERO-Pro.** For every task under each corresponding perturbation setting, we evaluate 10 seeds with 10 episodes per seed, totaling 100 rollouts per task-perturbation pair. Avg is the mean across the five perturbation settings. SFT baselines correspond to $P = 0$ and are reported once; "–" marks not applicable.

		Rollout budget $P = 100$						Rollout budget $P = 1000$					
Model	Policy opt.	Object	Position	Semantic	Task	Environment	Avg	Object	Position	Semantic	Task	Environment	Avg
OpenVLA	SFT	93	0	97	0	68	52	–	–	–	–	–	–
π_0	SFT	91	0	91	0	39	44	–	–	–	–	–	–
$\pi_{0.5}$	SFT	96	21	96	1	53	53	–	–	–	–	–	–
SureFlow	SFT	69	0	88	0	**88**	49	–	–	–	–	–	–
π_0	GRPO	90	0	90	0	30	42	92	10	92	8	44	49
π_0	DAPO	91	2	91	1	32	43	93	15	93	11	47	52
π_0	LPRO	92	31	94	23	49	58	93	42	95	32	55	63
$\pi_{0.5}$	GRPO	95	17	95	0	51	52	96	29	96	9	58	58
$\pi_{0.5}$	DAPO	95	18	95	2	54	53	96	34	96	14	61	60
$\pi_{0.5}$	LPRO	**97**	**49**	**98**	**36**	86	**73**	**97**	**58**	**98**	**46**	**92**	**78**



(a) Position Perturbation **(b)** Task Perturbation **(c)** Environment Perturbation

Figure 5: **Representative LIBERO-Pro Perturbation Settings. (a) Position Perturbation** relocates manipulable objects within feasible spatial bounds, illustrated by moving the cabinet, bowls, and cookie box. **(b) Task Perturbation** redefines task logic and target states: the depicted task requires two objects in the goal basket rather than one. **(c) Environment Perturbation** replaces the working environment to assess cross-environment robustness, represented by low illumination and visual noise. Yellow boxes indicate the goal basket (Zhou et al., 2025a).

Benchmark. We evaluate on LIBERO-PRO (Zhou et al., 2025a), an extension of LIBERO (Liu et al., 2023) with object, position, semantic, task, and environment evaluation suites. These suites test the central capabilities of DreamEvolve: appearance robustness, response to shifted spatial configurations, and transfer beyond the original task presentation. We select $\pi_{0.5}$ (Intelligence et al., 2025), OpenVLA (Kim et al., 2024), π_0 (Black et al., 2024), and SureFlow (Islam et al., 2026) as representative VLA policies for evaluation.

Post-training. DreamEvolve starts from the same LIBERO-SFT checkpoint as each baseline. For every task, it generates groups of successful, partial, and failed imagined rollouts. Appearance edit instructions modify object appearance, background, lighting, viewpoint, and non-interacting distractors while retaining the action-conditioned trajectory. Spatial edit instructions modify the manipulated-object or target position in the initial latent state; the policy then selects new actions in response to that state throughout the closed-loop imagined rollout. Both protocols preserve the task instruction and expose the policy to controlled variations. We evaluate rollout budgets of $P = 100$ and $P = 1000$ imagined rollouts per task and apply one round of post-training.

Results Table 1 shows a consistent LPRO advantage across the five evaluation suites. At $P = 100$, LPRO improves the average success rate by about 20 percentage points over GRPO and DAPO, and already surpasses their results with $P = 1000$, demonstrating substantially higher policy improvement per imagined rollout. At $P = 1000$, LPRO gains a further 5 percentage points while retaining its lead across suites, showing that DreamEvolve benefits both from low-budget adaptation and continued rollout scaling; larger imagined-rollout budgets are expected to yield further gains. Together, the cross-condition gains answer **Q1**: **DreamEvolve improves cross-scene and cross-task generalization and strengthens policy robustness under out-of-distribution shifts.** The cross-budget comparison answers **Q2**: **LPRO achieves more efficient policy improvement from each imagined rollout.**

5.2 REAL-WORLD ROBOT EXPERIMENTS

We further evaluate DreamEvolve on a single-arm Franka robot across three categories: *(1) Long-Horizon Compositional Manipulation*, *(2) Semantic and Spatial Reasoning*, and *(3) Visually Challenging Generalization*. The seven tasks require coherent multi-stage execution, object-scene reasoning, and control in cluttered or low-contrast scenes.

For each task, we collect 100 real-world demonstrations and fine-tune $\pi_{0.5}$ as the supervised baseline. DreamEvolve starts from this same checkpoint and applies one round of LPRO with 1000 imagined rollouts. We report success rate across real-world rollouts. Figure 6 shows representative task settings.

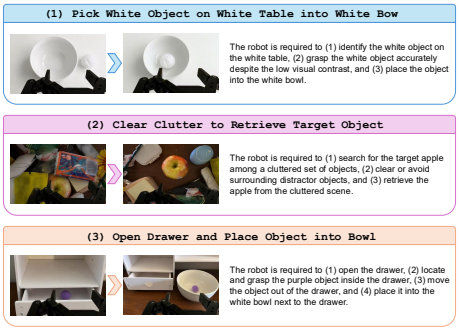


Figure 6: **Representative Real-World Evaluation Tasks.** Each row shows the initial scene and target state for a visually challenging or long-horizon Franka task.

Table 2: **Real-World Franka Success Rate (%).** C denotes clean scenes without perturbations, and P denotes perturbed scenes. DreamEvolve improves the $\pi_{0.5}$ SFT baseline in both settings. Each entry aggregates 100 physical evaluation rollouts.

	$\pi_{0.5}$		DreamEvolve	
Task	C	P	C	P
Open Drawer and Place Object into Bowl	13	*0*	**73**	*70*
Sort Objects by Color into Bowls	31	*2*	**86**	*82*
Wipe Stain with Cloth and Return Cloth	7	*0*	**54**	*32*
Open Laptop, Press Power, and Close Laptop	18	*1*	**69**	*65*
Stack Blocks from Large to Small	36	*3*	**74**	*71*
Clear Clutter to Retrieve Target Object	0	*0*	**48**	*25*
Pick White Object on White Table into White Bowl	45	*65*	72	***87***
Average	21.4	*10.1*	**68.0**	*61.7*

Table 2 shows consistent improvement across all seven tasks. On clean scenes, imagination-based post-training raises average success rate from 21.4% to 68.0%. Across deployment perturbations, including lighting, table color, camera viewpoint, object appearance, scene layout, etc., DreamEvolve attains 61.7% average success compared with 10.1% for the SFT baseline. The white-on-white task benefits from the perturbations because the altered scene introduces color contrast between the object, table, and bowl. These consistent physical gains answer **Q3**: **DreamEvolve delivers robust long-horizon real-world execution after a single round of latent imagination, retaining reliability across deployment perturbations.**

5.3 Latent World Model Evaluation

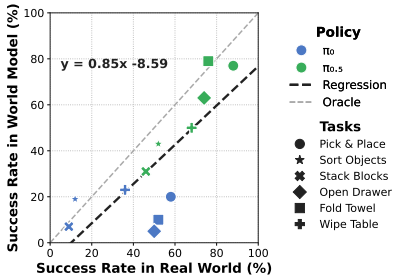


Figure 7: **Real-World Success Alignment.** World-model-predicted success aligns with success measured from real-robot rollouts across six representative tasks.

We evaluate the latent world model on a separate six-task real-robot suite, asking whether imagined rollouts preserve real-world policy ordering. Physical and imagined rollouts begin from the same observation, and the imagined branch receives the actions generated by each evaluated policy. Across six representative real-robot tasks, policies with greater physical task progress also receive greater latent progress. Localization errors, unstable contacts, and incomplete multi-stage executions produce correspondingly lower latent progress scores. Figure 7 visualizes the alignment between real-world success and world-model-predicted success. Across the 12 task-policy evaluations, world-model-predicted success achieves Pearson $r = 0.78$ with physical success and has a 17.8 percentage-point MAE. This agreement resolves **Q4**: **DreamEvolve's latent world model serves as a reliable policy evaluator, with predicted latent progress aligned with real-world task success.**

5.4 Ablation Study

To isolate the contribution of latent prediction and controllable scene variation, we conduct a four-way ablation on LIBERO-Pro. We compare pixel-level prediction, pixel-level prediction with scene variation, latent prediction, and latent prediction with scene variation. Each variant is post-trained from both π_0 and $\pi_{0.5}$ with $P = 1000$ imagined rollouts per task; results report the macro-average across the two backbones. In the pixel-level variants, the encoder is removed and the model directly predicts future video frames. This design directly tests whether latent imagination and controllable scene variation strengthen out-of-distribution policy improvement. We focus this ablation on Pos and Task, the most challenging LIBERO-Pro settings, following the evaluation settings of Harness-VLA and Zetta (Zhang et al., 2026; Ding et al., 2026). These perturbations require adaptation to shifted object configurations and unseen task compositions. As shown in Figure 8, latent prediction with scene variation performs best across both settings. The ablation answers **Q5**: **scene-variant generation can strengthen policy self-evolution, and DreamEvolve can achieve the best downstream task performance with latent prediction and scene-variant generation.**

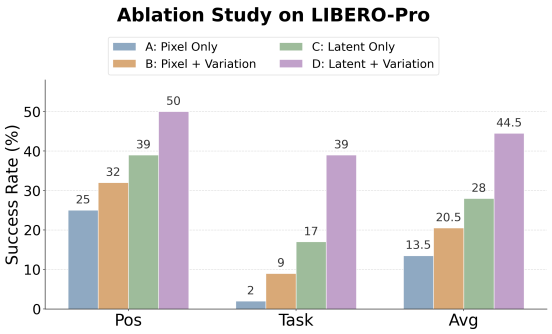


Figure 8: **LIBERO-Pro Ablation.** Macro-average success across π_0 and $\pi_{0.5}$ after post-training with $P = 1000$ imagined rollouts per task.

6 Conclusion

We presented DreamEvolve, a robotic self-evolution framework that uses controllable latent world models to improve generalist robot policies under distribution shifts. Across LIBERO-Pro and real-world Franka experiments, it improves robustness under cross-task, cross-scene, cluttered, low-contrast, and long-horizon conditions without additional human demonstrations. Latent imagination offers a scalable path toward adaptable robot policies.

AI USE STATEMENT

Generative AI tools were used only for editorial assistance, including improving language clarity, refining presentation, all AI-assisted text was reviewed and revised by the authors before inclusion in the manuscript.

ETHICS STATEMENT

DreamEvolve studies robotic policy learning and evaluation rather than human behavior. The physical experiments use a Franka robot on predefined tabletop manipulation tasks, with controlled scene perturbations, fixed rollout horizons, and standardized resets between trials. DreamEvolveBench is constructed from existing robot trajectories and generated scene variations, with automated filtering and deduplication applied during dataset construction. Our results are specific to the evaluated robot, tasks, and perturbation ranges, and deployment in substantially different physical settings should be separately validated.

REPRODUCIBILITY STATEMENT

The components required to reconstruct DreamEvolve are documented across the main paper and appendix. Section 3 describes the DreamEvolveBench generation and quality-control pipeline; Sections 4.1–4.3 define the latent world-model architecture, four-stage training procedure, and LPRO objective; and Section 5 specifies the simulation and physical-robot evaluation settings. Appendix A records optimization settings, stage-wise training schedules, loss weights, LPRO hyperparameters, and compute resources. Appendix B gives the full LPRO procedure in pseudocode, Appendix C reports DreamEvolveBench splits and filtering statistics, and Appendix D provides task-level success criteria, perturbation ranges, rollout limits, and reset procedures for the real-world experiments. We plan to release the implementation, training configurations, evaluation code, and DreamEvolveBench to support independent reproduction.

REFERENCES

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

Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, et al. Roboarena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 2025.

Qingyan Bai, Qiuyu Wang, Hao Ouyang, Yue Yu, Hanlin Wang, Wen Wang, Ka Leong Cheng, Shuailei Ma, Yanhong Zeng, Zichen Liu, et al. Scaling instruction-based video editing with a high-quality synthetic dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 37971–37981, 2026.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. Diwa: Diffusion policy adaptation with world models. *arXiv preprint arXiv:2508.03645*, 2025.

Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Topreward: Token probabilities as hidden zero-shot rewards for robotics. *arXiv preprint arXiv:2602.19313*, 2026.

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

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Xin Ding, Liang Mi, Mingzhe Huang, Zixuan Wang, Chao Zhang, Zixu Hao, Fu Chen, Xiangyu Li, Yikai Zheng, Yaoyu Guo, et al. Zetta ζ: An efficient closed-loop embodied harness for self-evolving physical intelligence. *arXiv preprint arXiv:2608.16590*, 2026.

Google DeepMind. Gemini 3.1 Pro model card. `https://deepmind.google/models/model-cards/gemini-3-1-pro/`, February 2026. Accessed: 2026-05-06.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 6121–6138, 2026.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. The franka emika robot: A reference platform for robotics research and education. *IEEE Robotics & Automation Magazine*, 29(2):46–64, 2022. doi: 10.1109/MRA.2021.3138382.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Yuhang Huang, Jiazhao Zhang, Shilong Zou, Xinwang Liu, Ruizhen Hu, and Kai Xu. Ladi-wm: A latent diffusion-based world model for predictive manipulation. *arXiv preprint arXiv:2505.11528*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Md Tanvir Islam, Sai Navaneet Peddapalli, Sangmoon Lee, and Sangtae Ahn. SureFlow: State-space uncertainty-aware residual flow matching for robust robot manipulation. *arXiv preprint arXiv:2607.10504*, 2026.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.

Zhennan Jiang, Kai Liu, Yuxin Qin, Shuai Tian, Yupeng Zheng, Mingcai Zhou, Chao Yu, Haoran Li, and Dongbin Zhao. World4rl: Diffusion world models for policy refinement with reinforcement learning for robotic manipulation. *arXiv preprint arXiv:2509.19080*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Yiqi Lin, Guoqiang Liang, Ziyun Zeng, Zechen Bai, Yanzhe Chen, and Mike Zheng Shou. Kiwi-edit: Versatile video editing via instruction and reference guidance. *arXiv preprint arXiv:2603.02175*, 2026.

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

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Jeremy Morgan, Hyeonho Oh, Prajwal Vijay, Jincen Song, Ashvin Arora, Hojung Lim, Alina Du, Jesse Thomason, Gaurav Sukhatme, and Ishika Singh. Colosseum v2: Benchmarking generalization for vision language action models. *arXiv preprint arXiv:2605.27759*, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

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, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on robot learning*, pp. 1723–1736. PMLR, 2023.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pp. 2226–2240. PMLR, 2023.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Yi Nie, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026.

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

Zhiyuan Zhou, Pranav Atreya, You Liang Tan, Karl Pertsch, and Sergey Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. *arXiv preprint arXiv:2503.24278*, 2025b.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: A fine-grained world model for robot manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9834–9844. IEEE, 2025.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. Wmpo: World model-based policy optimization for vision-language-action models. In *International Conference on Learning Representations*, volume 2026, pp. 62486–62502, 2026.

APPENDIX

APPENDIX OVERVIEW

SECTION	
§A	**Implementation & Training Details:** We specify the optimizer, learning rate, batch size, four-stage training steps, LoRA configuration, loss weights, LPRO hyperparameters, and compute infrastructure used by DreamEvolve.
§B	**LPRO Algorithm and Pseudocode:** We present complete LPRO pseudocode from closed-loop latent rollouts and latent progress rewards through group normalization, time-aligned advantages, and policy updates.
§C	**DreamEvolveBench Details:** We report source-disjoint dataset splits, edit-category proportions, and quality-control acceptance statistics.
§D	**Real-World Evaluation Protocol:** We define task success criteria, initialization, perturbation ranges, timeouts, reset procedures, and per-task breakdowns for real-world evaluation.

A IMPLEMENTATION & TRAINING DETAILS

A.1 OPTIMIZATION CONFIGURATION

Table 3: **Optimization Configuration.** All stages use AdamW with bfloat16 training.

Setting	Value
Optimizer	AdamW
AdamW coefficients	$\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay $= 0.05$
Global batch size	256 video clips
Microbatch and accumulation	1 clip per GPU, accumulation $4/4/8/16$ across Stages 1-4
Learning-rate warmup	Linear warmup over the first 5% of each stage
Precision and gradient clipping	bfloat16, global gradient norm 1.0

Latent-Edit Adapter. The frozen Qwen3.5-4B backbone is adapted with LoRA of rank 64, scaling factor 128, and dropout 0.05 on attention and feed-forward linear projections. These adapter settings are separate from the shared optimizer configuration in Table 3.

A.2 FOUR-STAGE TRAINING RECIPE

Table 4: **Four-Stage Training Schedule.** The schedule reduces the learning rate as supervision becomes more robot-specific.

Stage	Training source	Updates	Peak learning rate	GPUs $\times$ hours
1	Action-free web videos with a null-action token	200K	1×10^{-4}	64×235
2	Egocentric videos with a null-action token	100K	8×10^{-5}	64×115
3	Generated heterogeneous robot trajectory variants	50K	5×10^{-5}	32×113
4	DROID trajectory variants for the Franka target robot	20K	2×10^{-5}	16×89

A.3 Loss Weights and LPRO Hyperparameters

Table 5: **Loss Weights and LPRO Hyperparameters.** The asymmetric clipping bounds follow the DAPO-style policy update used by LPRO.

Setting	Value
Future visual-latent rollout weight λ_{roll}	1.0
Masked-action reconstruction weight λ_{act}	0.1
Imagined-latent distillation weight λ_{imag}	0.5
Progress-to-go discount factor γ	0.99
Group-normalization stabilizer ϵ	10^{-6}
Lower clipping bound ϵ_{low}	0.20
Upper clipping bound ϵ_{high}	0.28
Policy learning rate	1×10^{-5}

A.4 Compute Infrastructure

World-model training uses H100 80GB GPUs with FSDP sharding, FlashAttention, and activation checkpointing. The four training stages consume 27,440 GPU-hours in total. LPRO post-training uses eight H100 80GB GPUs for each policy backbone.

B LPRO Algorithm and Pseudocode

B.1 Closed-Loop Latent Rollouts

Notation. π_θ^{pv} denotes the trainable policy over visual latents, with parameters θ. W_ϕ denotes the fixed action-conditioned latent world model, and P_η denotes the fixed latent progress model. The instruction is l, the initial policy visual latent is h_0, and X_{cond} is either the edited-state conditioning context or the null context. A rollout group has G trajectories, each with action horizon H. The target-chunk length is K, where K divides H, and c indexes a target chunk with starting action position $t = cK$. The discount factor is γ, the normalization stabilizer is ϵ, and ϵ_{low} and ϵ_{high} are the lower and upper policy-ratio clipping bounds. For trajectory i and action step t, $\hat{a}_{i,t}$ and $\hat{h}_{i,t}$ are the sampled action and imagined latent state. $\hat{\mathbf{a}}_{i,t+1:t+K}$ denotes the policy action chunk sampled autoregressively from the current latent context, and $\hat{\mathbf{h}}_{i,t+1:t+K}$ denotes the resulting world-model target chunk. $p_{i,t}$ is the predicted progress, with $p_{i,0} = P_\eta(\hat{h}_{i,0}, l)$, $d_{i,t}$ is the stop-gradient progress increment, and $g_{i,t}$ is the discounted progress-to-go. The operator sg blocks gradients through progress differences. μ_t and σ_t are the across-group mean and standard deviation of $g_{i,t}$ at step t, $\hat{A}^{\Delta}_{i,t}$ is the resulting time-aligned advantage, and $\mathcal{T}$ retains rollout positions with non-degenerate group variation. $\mathrm{MeanStd}$ computes the stated group statistics, and $\mathrm{AsymClipFPO}$ denotes the asymmetric clipped flow-policy update described in the main paper.

Algorithm 1: Latent Progress Reward Optimization (LPRO)

Input: π_θ^{pv}, W_ϕ, P_η, l, h_0, X_{cond}, rollout group size G, action horizon H, target-chunk length K, γ, ϵ, ϵ_{low}, and ϵ_{high}.
for $i \leftarrow 1$ **to** G **do**
  $\hat{h}_{i,0} \leftarrow h_0$
  $p_{i,0} \leftarrow P_\eta(\hat{h}_{i,0}, l)$
  for $c \leftarrow 0$ **to** $H/K - 1$ **do**
    $t \leftarrow cK$
    $\hat{\mathbf{a}}_{i,t+1:t+K} \sim \pi_\theta^{\mathrm{pv}}(\hat{h}_{i,0:t}, l)$
    $\hat{\mathbf{h}}_{i,t+1:t+K} \leftarrow W_\phi(\hat{h}_{i,0:t}, \hat{\mathbf{a}}_{i,t+1:t+K}, X_{\mathrm{cond}}, l)$
    $p_{i,t+j} \leftarrow P_\eta(\hat{h}_{i,0:t+j}, l)$ for every $j \in \{1, \ldots, K\}$
    $d_{i,t+j} \leftarrow \mathrm{sg}(p_{i,t+j} - p_{i,t+j-1})$ for every $j \in \{1, \ldots, K\}$
  end for
  $g_{i,t} \leftarrow \sum_{k=t}^{H} \gamma^{k-t} d_{i,k}$ for every t
end for
$\mu_t, \sigma_t \leftarrow \mathrm{MeanStd}_{i=1}^{G}(g_{i,t})$ for every t
$\hat{A}_{i,t}^{\Delta} \leftarrow (g_{i,t} - \mu_t)/(\sigma_t + \epsilon)$ for every i, t
$\mathcal{T} \leftarrow \{t \in \{1, \ldots, H\} : \sigma_t > \epsilon\}$
$\theta \leftarrow \mathrm{AsymClipFPO}(\theta, \{\hat{A}_{i,t}^{\Delta} : i \in \{1, \ldots, G\}, t \in \mathcal{T}\}, \epsilon_{\mathrm{low}}, \epsilon_{\mathrm{high}})$
Output: updated policy parameters θ

C DreamEvolveBench Details

C.1 Dataset Statistics

Table 6: **DreamEvolveBench Statistics.** Global edits use robot-removed background references, while local object edits use grounded object-centric references.

Statistic	Variants	Share or retention
Source-disjoint split		
Train	902,817	90.3%
Validation	49,375	4.9%
Test	47,808	4.8%
Edit distribution		
Background	257,707	25.8%
Surrounding objects	201,580	20.2%
Manipulated object	198,857	19.9%
Lighting	145,923	14.6%
Table material	98,510	9.9%
Viewpoint	97,423	9.7%
Quality control		
Generated edited-video candidates	1,575,100	100%
Instruction-aligned and temporally consistent candidates	1,119,325	71.06%
Reference-aligned candidates after deduplication	1,000,000	63.49%

D Real-World Evaluation Protocol

D.1 Task Success Criteria

Table 7: **Real-World Task Protocol.** Each task evaluates the stated terminal state under both clean and perturbed initial conditions.

Task	Terminal success state	Primary category
Open Drawer and Place Object into Bowl	The drawer is open and the target object is inside the bowl.	Long-horizon manipulation
Sort Objects by Color into Bowls	Each object is placed in the bowl with the matching color.	Semantic reasoning
Wipe Stain with Cloth and Return Cloth	The stain is wiped and the cloth is returned to the designated location.	Long-horizon manipulation
Open Laptop, Press Power, and Close Laptop	The laptop is opened, the power control is pressed, and the laptop is closed.	Long-horizon manipulation
Stack Blocks from Large to Small	Blocks form a stable stack in descending size order.	Spatial reasoning
Clear Clutter to Retrieve Target Object	Clutter is cleared and the target object is retrieved.	Semantic and spatial reasoning
Pick White Object on White Table into White Bowl	The white target object is placed in the white bowl.	Visually challenging generalization

D.2 Initialization and Perturbation Ranges

Every trial begins with the Franka robot at a fixed home configuration and the task objects placed in a canonical scene arrangement. Perturbed trials apply one or more deployment variations from Table 8 before the policy receives the initial observation.

Table 8: **Real-World Perturbation Protocol.** The clean condition uses the canonical scene, while the perturbed condition varies visual and spatial factors within the listed ranges.

Perturbation	Range or setting
Illumination	300 to 800 lux, including directional lighting changes
Table appearance	White, black, and light-wood table surfaces
Camera viewpoint	Translation up to 5 cm, yaw up to $\pm 15^{\circ}$, pitch up to $\pm 10^{\circ}$
Object appearance	Color and texture changes that preserve object geometry
Scene layout	Non-target object relocation up to 5 cm

D.3 Timeout and Reset Protocol

Each real-world rollout has a maximum horizon of 300 control steps at 10 Hz. After every rollout, the robot returns to the home configuration, all task objects return to the prescribed initial locations, and the camera calibration is checked before the next evaluation.