WCM: A WORLD CRITIC MODEL FOR VISION-LANGUAGE-ACTION REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) can improve Vision-Language-Action (VLA) policies beyond supervised fine-tuning, but critic-based methods depend on value estimates that must account for partial observability. Critics operating on current-observation features can miss temporal information relevant to task progress, while simply adding observation history does not reliably improve policy performance in our experiments. We propose the **World Critic Model (WCM)**, a history-conditioned critic that combines value estimation with action-conditioned latent prediction. Built on a lightweight LeJEPA-based architecture, WCM encodes a short observation history and a language instruction into a shared temporal representation, from which it estimates returns and predicts the next observation latent under the executed action. The prediction objective complements return supervision by encouraging the representation to retain transition-relevant information, without requiring pixel reconstruction or imagined rollouts for value estimation. WCM integrates into both on-policy and off-policy RL pipelines and supports diverse VLA backbones. Experiments spanning 149 tasks across four simulation benchmarks demonstrate improvements in policy performance and generalization, with ablations supporting the benefit of combining history with latent prediction. Evaluations on seven real-world manipulation tasks further demonstrate the effectiveness of WCM in off-policy VLA reinforcement learning.

1 INTRODUCTION

Vision-Language-Action (VLA) models have demonstrated strong performance in robotic manipulation (Black et al., 2024; Intelligence et al., 2025b; 2026; Kim et al., 2025; 2024). While supervised fine-tuning (SFT) learns from a fixed collection of expert demonstrations, reinforcement learning (RL) can further improve these policies through environment interaction (Liu et al., 2025b; Fei et al., 2025a; Li et al., 2025b; Zang et al., 2025). In critic-based VLA-RL, value estimates help translate trajectory outcomes into training signals that weight or guide policy updates (Intelligence et al., 2025a; Chen et al., 2025; Zang et al., 2026). The quality of the critic therefore matters not only for predicting returns, but also for the policy improvement driven by those predictions.

A central challenge for critic learning is partial observability. A current image can reveal scene layout and object appearance while leaving motion and interaction progress ambiguous. For example, similar gripper configurations may occur while approaching an object or withdrawing after an unsuccessful grasp; recent observations can help distinguish these situations, which may have different expected returns. In a POMDP, a sufficient statistic of observation-action history supports value estimation when the current observation alone is insufficient (Littman & Sutton, 2001; Singh et al., 2003; Subramanian et al., 2022). However, representative VLA-RL critics operate primarily on current-observation features, either from a separate visual encoder or from the policy's VLM backbone (Lu et al., 2025; Liu et al., 2025b; Zang et al., 2025; Chen et al., 2025). Such critics may consequently conflate visually similar situations that differ in their temporal context.

Providing observation history is a natural response, but making temporal information available does not ensure that critic learning will exploit it. In our experiments, augmenting value-only critics with stacked observations or a temporal Transformer does not reliably improve policy performance as history length increases. We hypothesize that return regression alone provides an insufficient inductive bias for retaining transition-relevant information in these settings: it supervises the expected

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.

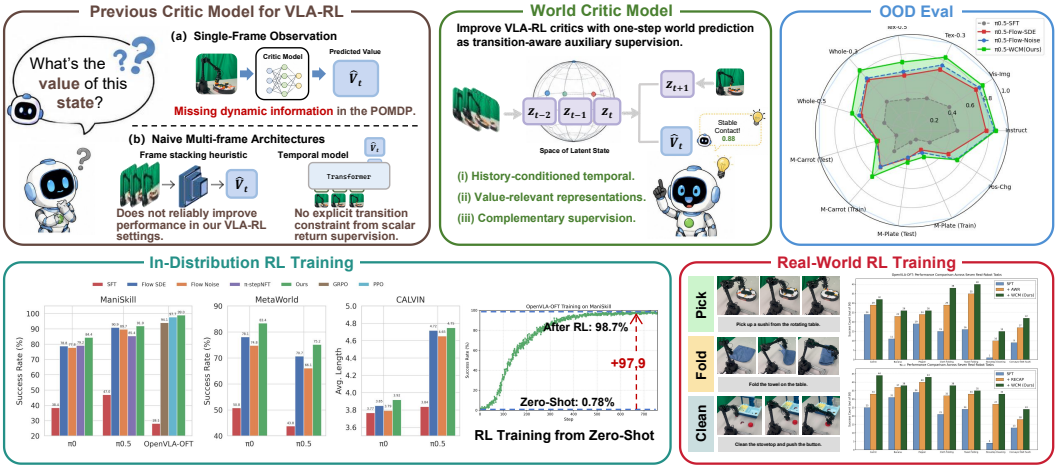


Figure 1: Overview of the World Critic Model (WCM). Current observations can leave task progress ambiguous, motivating the use of temporal context for value estimation. WCM combines a short observation history and a task instruction into a shared representation for value estimation and action-conditioned next-latent prediction. The prediction objective complements return supervision during critic training. WCM integrates into both on-policy and off-policy VLA-RL pipelines.

outcome, but does not explicitly require the representation to predict how observations evolve under the executed action. This motivates a complementary training objective that encourages the critic to learn from observed transitions as well as trajectory returns.

Predictive representation learning offers a way to introduce this objective. Prior work in visual RL uses latent transition prediction to learn representations useful for downstream control (Gelada et al., 2019; Guo et al., 2020; Schwarzer et al., 2020). We build on this principle to study history-conditioned critic learning for VLA-RL. Our focus is to couple action-conditioned prediction and value estimation through a shared temporal representation, conditioned on the task instruction. Predicting in latent space provides transition supervision without requiring pixel reconstruction, while joint value learning encourages the representation to remain useful for estimating task returns.

We propose the **World Critic Model (WCM)**, a predictive critic built on a lightweight LeJEPA-based architecture (Balestriero & LeCun, 2025). WCM encodes a short observation history and the language instruction into a shared temporal representation. A value head estimates the expected return from this representation, while a separate prediction head additionally conditions on the executed action to predict the next observation latent. Joint optimization encourages the shared representation to retain information useful for both value estimation and transition prediction. WCM can replace the critic in both on-policy and off-policy VLA-RL pipelines, without requiring imagined future rollouts to compute its value estimates.

We evaluate WCM across 149 tasks from four simulation benchmarks using π_0, $\pi_{0.5}$, and OpenVLA-OFT. On ManiSkill, replacing the Flow-SDE critic with WCM improves average OOD success by 12.2 and 15.1 percentage points for π_0 and $\pi_{0.5}$, respectively. Experiments on Meta-World, CALVIN, and LIBERO-Plus further evaluate performance across diverse manipulation tasks, long-horizon execution, and distribution shifts. On seven real-world tasks, WCM yields higher observed success rates than the baseline critics in both AWR and RECAP. Ablations across critic architectures and history lengths support the benefit of combining temporal context with the prediction objective.

Our contributions are threefold:

1. We empirically study history conditioning in VLA-RL critics and show that additional observations do not reliably improve the tested value-only critics, motivating supervision beyond return regression.
2. We introduce WCM, which jointly learns value estimation and action-conditioned next-latent prediction through a shared language-conditioned temporal representation.

3. We evaluate WCM across three VLA backbones, four simulation benchmarks, and seven real-world tasks, demonstrating its applicability to both on-policy and off-policy RL and improvements over the corresponding baseline critics.

2 Related Work

Critic models in Vision-Language-Action Reinforcement Learning (VLA-RL) suffer from partial observability. VLA (Driess et al., 2023; Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024; Pertsch et al., 2025; Intelligence et al., 2025a;b; 2026) models have emerged as a promising paradigm in robot manipulation. The reliance on expert demonstrations (Intelligence et al., 2025a; Fei et al., 2025a; Zang et al., 2025; Li et al., 2025b) of supervised fine-tuning has driven a growing number of studies toward RL post-training, as it offers better performance (Liu et al., 2025b; Li et al., 2025b; Fei et al., 2025a) and smoother deployment (Zang et al., 2025; Chen et al., 2025; Intelligence et al., 2025a). Recent work has extended reinforcement-learning post-training to both autoregressive (Schulman et al., 2017; Liu et al., 2024; Haarnoja et al., 2018; Kostrikov et al., 2021; Peng et al., 2019) and flow-matching (Zhang et al., 2025; Liu et al., 2025a; Chen et al., 2025; Ren et al., 2024) VLA architectures. We focus on the critic-based subset of these methods. On-policy approaches (Wagenmaker et al., 2025; Lu et al., 2025; Zang et al., 2025; Chen et al., 2025) use state-value to construct stepwise advantage estimates, whereas off-policy approaches (Luo et al., 2025; Intelligence et al., 2025a; Peng et al., 2019; Team et al., 2026) use value-derived advantages to weight action likelihoods or condition policy learning. Recent VLA-RL research has focused primarily on policy optimization, flow or diffusion likelihoods, and systems scaling (Intelligence et al., 2025a; Lu et al., 2025; Zang et al., 2025), while representative critics still estimate value mainly from a current-step observation representation. WCM studies a question orthogonal to policy parameterization: how a VLA-RL critic should represent temporal state when the current observation is value-ambiguous, as single frames may lose global or dynamic information (Littman & Sutton, 2001; Singh et al., 2003; Laskin et al., 2020; Jiang et al., 2017).

Directly applying existing history-conditioning mechanisms to VLA-RL critics introduces computational or supervision challenges. Observation frame stacking, recurrent states, and predictive-state representations have long been used to address partial observability in reinforcement learning (Mnih et al., 2015; Hausknecht & Stone, 2015; Littman & Sutton, 2001; Singh et al., 2003; Liu et al., 2022). Recent VLA methods similarly incorporate memory or multiple frames into the policy to improve action prediction under temporal ambiguity (Shi et al., 2025; Koo et al., 2025; Li et al., 2025a). Directly transferring these mechanisms to a VLA-RL critic, however, presents both computational and supervision challenges. Frame stacking expands the high-dimensional visual input with the history length and repeatedly processes appearance shared across adjacent frames (Shang et al., 2021; Efroni et al., 2022). A recurrent critic can compress history into a hidden state, but introduces sequential dependencies across time, as training requires sequence unrolling and backpropagation through time (BPTT) (Martin & Cundy, 2017; MacKay et al., 2018). Replay-based training additionally requires careful hidden-state initialization, stale-state handling, or burn-in (Gruslys et al., 2016; Kapturowski et al., 2018). A feed-forward temporal encoder can aggregate a fixed short window in parallel, but when its representation is learned only from return-derived scalar targets, it still receives no explicit constraint on how the executed action transforms the observation history (Yarats et al., 2021; Ahuja et al., 2022). Existing history-aware VLA designs either require memory-state reconstruction during minibatch updates or provide no explicit executed-transition constraint on the value representation, making them nontrivial to use for frequent critic learning (Shi et al., 2025; Koo et al., 2025; Li et al., 2025a). WCM therefore leverages a short-window feed-forward history trunk and couples next-latent prediction to value learning, avoiding recurrent unrolling and placing a transition-aligned predictive constraint on the critic state.

WCM bears some resemblance to DeepMDP and SPR, which also use action-conditioned latent transition prediction to shape representations for reinforcement learning (Gelada et al., 2019; Schwarzer et al., 2020). DeepMDP combines latent transition and reward prediction with observation reconstruction and separate encoders, while SPR regularizes an Atari Q-network using a fixed frame stack, multi-step latent rollouts, and an exponential-moving-average (EMA) target encoder. Our contribution is to couple action-conditioned next-latent prediction with a language- and history-conditioned VLA-RL critic and to show, through matched history-only controls, that transition-aware supervision changes whether temporal context improves value estimation and policy updating. The prediction objective itself follows established principles and is not claimed as a new technique.

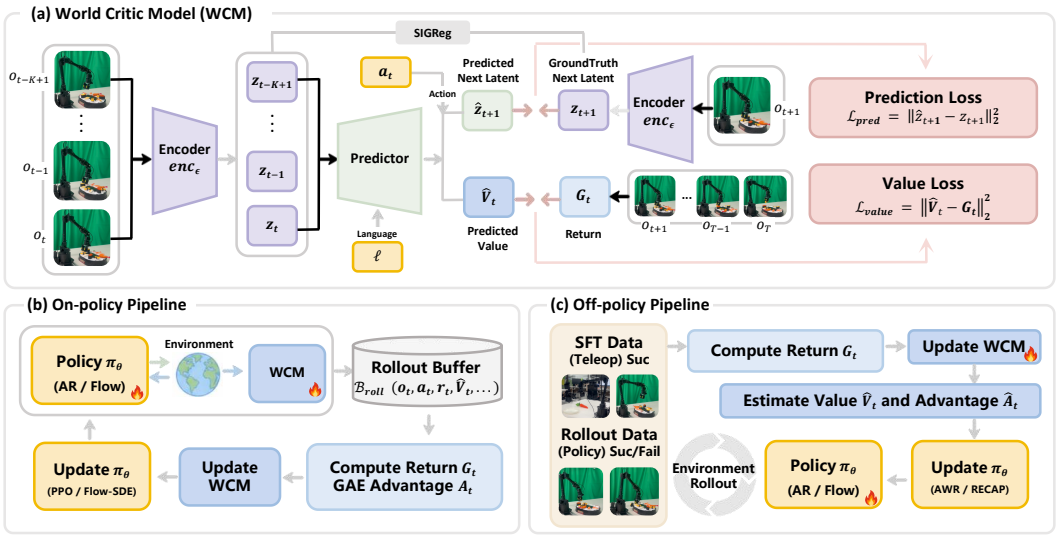


Figure 2: Overview of the World Critic Model (WCM) method. (a) **Architecture.** An observation encoder first maps individual observations from the past K frames into sequential latent space. Concurrently, a predictor and two decoder heads forecasts the next latent and estimates the value respectively. (b) **On-policy Pipeline**: WCM estimates values from on-policy rollouts to compute GAE advantages for PPO/Flow-SDE updates. (c) **Off-policy Pipeline**: Unified data buffers (SFT + rollouts) are utilized to stably update both WCM and the policy via AWR/RECAP.

3 Methodology

We consider VLA control under partial observability. At timestep t, a VLA policy π_θ receives an observation o_t and task instruction ℓ, and executes an action a_t, with the objective of maximizing the expected discounted return. Because a single observation may not uniquely identify task progress, value estimation can benefit from temporal context. Rather than attempting to construct a sufficient statistic of the underlying POMDP, we focus on the practical value of short observation histories for critic learning in VLA reinforcement learning. We therefore condition the critic on a short observation history $H_t = (o_{t-K+1}, \ldots, o_{t-1}, o_t)$ together with ℓ. WCM learns a shared temporal representation for value estimation and action-conditioned prediction of the next observation latent. The latter provides transition-aware auxiliary supervision.

3.1 World Critic Model

To unify next-latent prediction and value estimation within a single framework while minimizing model complexity to enable training from scratch, we adopt the lightweight LeJEPA (Balestriero & LeCun, 2025; Maes et al., 2026) architecture as the basis of our World Critic Model (WCM).

Model Architecture. Our model consists of four components: an observation encoder, a temporal history trunk, and two lightweight heads. Given an observation history $H_t = (o_{t-K+1}, \ldots, o_t)$ and a language instruction ℓ, WCM constructs a shared language-conditioned temporal representation:

$$\boldsymbol{h}_t = \mathrm{Tr}_\phi(\mathrm{XAttn}\,(\boldsymbol{z}_{t-K+1:t}, \mathbf{u}_\ell)) \in \mathbb{R}^d, \tag{1}$$

where $\boldsymbol{z}_{t-K+1:t}$ denotes the sequence of per-frame visual representations, $\mathbf{u}_\ell$ denotes the language representation, XAttn conditions the visual sequence on the language representation, and Tr_ϕ is a causal Transformer history trunk.

Concretely, each observation in the history is independently processed by a shared observation encoder enc_ϵ:

$$\boldsymbol{z}_{t-k} = \mathrm{enc}_\epsilon(o_{t-k}), \qquad \forall k \in \{0, 1, \ldots, K-1\}. \tag{2}$$

Thus, $\boldsymbol{z}_{t-K+1:t} = (\boldsymbol{z}_{t-K+1}, \ldots, \boldsymbol{z}_t)$ forms the visual sequence used in Eq. 1. The language instruction ℓ is encoded by a pretrained CLIP (Radford et al., 2021) text encoder and projected into

the WCM latent space through a learned adapter $\mathcal{A}_{\text{lang}}$:

$$\mathbf{u}_\ell = \mathcal{A}_{\text{lang}}(\text{CLIP}(\ell)) \in \mathbb{R}^d. \tag{3}$$

The shared representation $\boldsymbol{h}_t$ is subsequently fed into two separate decoder heads: a value decoder head $\mathcal{D}_{\text{value}}$ for return estimation. The value estimation is given by

$$\hat{V}_t = \mathcal{D}_{\text{value}}(\boldsymbol{h}_t) \ \in \ \mathbb{R}, \tag{4}$$

and the action-conditioned branch predicts the next latent using a residual update:

$$\hat{\boldsymbol{z}}_{t+1} = \mathcal{D}_{\text{world}}(\boldsymbol{h}_t, a_t, \boldsymbol{z}_t) \in \mathbb{R}^d, \tag{5}$$

where the one-step world-prediction head $\mathcal{D}_{\text{world}}$ that predicts the next latent is implemented with an action encoder and gated FiLM (Perez et al., 2018) residual blocks.

Training Objective. The overall training objective combines three components: the prediction loss, the Sketched-Isotropic Gaussian Regularization (SIGReg) (Balestriero & LeCun, 2025) loss, and the value estimation loss. The **prediction loss** $\mathcal{L}_{\text{pred}}$ computes the error between the predicted next latent and the true next latent:

$$\mathcal{L}_{\text{pred}} = \|\hat{\boldsymbol{z}}_{t+1} - \boldsymbol{z}_{t+1}\|_2^2, \tag{6}$$

where $\hat{\boldsymbol{z}}_{t+1}$ is the predicted latent and $\boldsymbol{z}_{t+1}$ is the ground truth. The target latent z_{t+1} is used only to construct the auxiliary prediction loss during training. No future observation is fed into h_t.

To prevent feature collapse in the joint encoder–predictor objective, we apply Sketched-Isotropic Gaussian Regularization (SIGReg) (Balestriero & LeCun, 2025) to the latent representations $\boldsymbol{z}$, encouraging them to match an isotropic Gaussian distribution. The **SIGReg loss** is defined as:

$$\mathcal{L}_{\text{SIGReg}} = \mathbb{E}_{\mathbf{a}\sim\mathcal{U}(\mathcal{S}^{d-1})}\left[\int_{\mathbb{R}} \left|\hat{\phi}_{\mathbf{a}^\top \boldsymbol{z}}(t) - \phi(t)\right|^2 e^{-t^2}\, dt\right], \tag{7}$$

where $\mathbf{a}$ is a random unit vector drawn uniformly from the $(d-1)$-sphere, $\phi(t) = e^{-t^2/2}$ is the characteristic function of a standard normal, and $\hat{\phi}_{\mathbf{a}^\top \boldsymbol{z}}(t) = \mathbb{E}_{\boldsymbol{z}}\left[e^{it(\mathbf{a}^\top \boldsymbol{z})}\right]$ is the empirical characteristic function of the projected representation, approximated over a mini-batch. Intuitively, SIGReg enforces that every one-dimensional projection of $\boldsymbol{z}$ onto a random unit vector $\mathbf{a}$ matches the characteristic function of a standard Gaussian, which is a necessary and sufficient condition for $\boldsymbol{z}$ to follow an isotropic Gaussian distribution. SIGReg therefore prevents dimensional collapse and mode degeneration, while allowing the source and target embeddings to be learned jointly, removing the need for the separate EMA target encoder used in many other JEPA formulations.

The **value loss** $\mathcal{L}_{\text{value}}$ aims to predict the return associated with each state. For each time step in the trajectory, the reward r_t and return G_t are given as:

$$r_t = \begin{cases} 1 & \text{if } t = T \text{ and success,} \\ 0 & \text{otherwise,} \end{cases} \qquad G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}, \tag{8}$$

where T is the last step in the episode, and γ is the discount factor. For each task, we min-max normalize the returns to be in $[-1, 1]$ in that task's training trajectories. The value loss $\mathcal{L}_{\text{value}}$ is the L2 loss between the predicted value $\hat{V}_t$ and the ground truth return G_t:

$$\mathcal{L}_{\text{value}} = \|\hat{V}_t - G_t\|_2^2. \tag{9}$$

The **complete training objective** is then:

$$\mathcal{L} = \mathcal{L}_{\text{value}} + \lambda \cdot \mathcal{L}_{\text{pred}} + \eta \cdot \mathcal{L}_{\text{SIGReg}}, \tag{10}$$

where λ is a hyperparameter controlling the weight of the prediction loss, and η is a hyperparameter controlling the weight of the SIGReg regularization.

3.2 Training Pipeline

On-policy Setting: For Auto-Regressive (AR) models such as OpenVLA-OFT (Kim et al., 2025), we employ the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm. For flow-matching models like π_0 (Black et al., 2024) and $\pi_{0.5}$ (Intelligence et al., 2025b), we utilize Flow-SDE (Chen et al., 2025), a variant of PPO. In both frameworks, we adopt WCM as the critic model.

Table 1: Performance comparison on ManiSkill under IND and OOD settings. All methods are trained via RL on top of SFT initialization. For π_0 and $\pi_{0.5}$, "+WCM" denotes replacing the critic model with WCM on Flow-SDE. For OpenVLA-OFT, "+WCM" denotes the replacement on PPO.

Backbone	Method	IND		OOD				
		avg.	Δ	vision	semantic	execution	avg.	Δ
π_0	SFT	38.4	-	32.6	8.4	13.2	18.1	-
	+ FlowSDE	78.8	+40.4	61.1	25.4	31.5	39.3	+21.2
	+ FlowNoise	77.8	+39.4	63.4	23.1	24.2	36.9	+18.8
	+ π-stepNFT	79.2	+40.8	**69.1**	49.1	33.1	50.4	+32.3
	+ WCM (Ours)	**89.9**±0.7	**+51.5**±0.7	**69.1**±0.7	**49.8**±2.2	**35.6**±1.3	**51.5**±1.5	**+33.4**±1.5
$\pi_{0.5}$	SFT	40.1	-	40.2	16.6	22.4	26.4	-
	+ FlowSDE	90.9	+50.8	68.0	34.5	45.4	49.3	+22.9
	+ FlowNoise	89.7	+49.6	69.9	35.5	54.9	53.4	+27.0
	+ π-stepNFT	85.4	+45.3	76.9	56.6	45.1	59.5	+33.1
	+ WCM (Ours)	**95.1**±0.6	**+55.0**±0.6	**78.1**±1.6	**58.5**±1.0	**56.5**±1.5	**64.4**±1.4	**+38.0**±1.4
OpenVLA-OFT	SFT	28.1	-	27.7	13.0	11.7	18.3	-
	+ GRPO	94.1	+66.0	84.7	45.5	44.7	60.6	+42.3
	+ PPO	97.7	+69.6	92.1	64.8	73.6	77.1	+58.8
	+ WCM (Ours)	**99.0**±0.4	**+70.9**±0.4	**92.4**±0.5	**65.9**±1.0	**75.5**±0.7	**77.9**±0.8	**+59.6**±0.8
	Zero-Shot	0.8	-	0.5	1.0	1.0	0.8	-
	+ WCM (Ours)	98.7±0.3	+97.9±0.3	88.0±1.2	62.4±2.9	70.1±0.3	73.5±1.8	+72.7±1.8

To leverage the representations already learned during pretraining, we use the VLM backbone latent as the observation input to WCM. The detailed algorithm is outlined in Appendix D Algorithm 1.

Off-policy Setting: In the off-policy setting, each training iteration incorporates not only SFT data collected via teleoperation but also data from erroneous rollouts and typical failure cases. (Rollout data is unavailable in the first iteration.) These additional data expose the critic to both successful and failed outcomes during training. For AR models, we adopt the Advantage-Weighted Regression (AWR) (Peng et al., 2019). For flow-matching models, we use the RECAP (Intelligence et al., 2025a) method introduced by $\pi^*_{0.6}$. In both cases, we instantiate the critic model with WCM. The algorithms are shown in Appendix D Algorithms 2 and 3.

4 Experiments

We organize our experiments around six questions. (1) Does replacing existing VLA-RL critics with WCM improve policy performance? (2) Does WCM retain its advantage under distribution shifts? (3) Can WCM be effectively used for real-world off-policy RL? (4) Does next-latent prediction help the critic exploit observation history? (5) How does WCM compare with alternative history-aware and distributional critics? (6) How does performance vary with the observation-history length?

4.1 Experimental Setup

Simulation. We evaluate our method on 149 tasks across four simulation benchmarks. Following the RL4VLA (Liu et al., 2025b) setup, we leverage ManiSkill (Mu et al., 2021) and assess both in-distribution (IND) and out-of-distribution (OOD) performance across three axes. Additionally, we evaluate on MetaWorld (Yu et al., 2020) for performance on tasks beyond pick-and-place, CALVIN (Mees et al., 2022) for long-horizon capabilities, and LIBERO-Plus (Fei et al., 2025b) for generalization abilities. All policies are initialized from few-shot SFT baselines and trained using sparse 0/1 rewards, including π_0 (Black et al., 2024), $\pi_{0.5}$ (Intelligence et al., 2025b), and OpenVLA-OFT (Kim et al., 2025).

Real-world. We evaluate our method on 7 robotic tasks on WidowX-250S: a dynamic grasping task, a long-horizon task, 2 deformable object manipulation tasks, and 3 pick-and-place tasks. Using $\pi_{0.5}$ and OpenVLA-OFT as base policies, we train them with off-policy RL guided by our WCM.

Table 2: Detailed performance on LIBERO-Plus. "One-SFT" is the SFT model seeing only one demonstration per task, while "Full-SFT" seeing 50. "+WCM" is initialized from One-SFT.

Backbone	Method	Metrics							
		camera	env	init	language	noise	layout	light	total
π_0	Full-SFT	85.4±1.4	90.1±1.8	12.5±0.9	65.8±1.3	**89.7**±2.5	75.3±1.2	89.3±0.7	71.2±1.5
	One-SFT	32.3±2.2	59.4±0.2	1.5±0.9	42.3±1.7	48.7±0.4	50.2±2.8	48.0±3.8	39.1±2.1
	+ WCM (Ours)	**88.3**±2.7	**91.0**±1.8	**19.3**±1.0	**65.9**±0.2	87.5±0.2	**76.0**±1.4	**90.9**±3.6	**72.8**±1.9
	Δ	+56.0	+31.6	+17.8	+23.6	+38.8	+25.8	+42.9	+33.7
$\pi_{0.5}$	Full-SFT	78.8±0.5	89.9±2.9	24.7±1.2	73.8±1.0	**88.2**±1.7	77.0±2.0	78.3±1.9	72.9±1.8
	One-SFT	32.8±0.5	52.7±3.0	1.9±0.6	38.7±0.7	50.5±2.2	46.1±1.5	51.2±0.6	38.0±1.6
	+ WCM (Ours)	**80.8**±1.7	**90.3**±1.6	**31.5**±0.6	**65.0**±1.1	86.3±2.1	**79.3**±0.9	**91.6**±0.8	**73.7**±1.4
	Δ	+48.0	+37.6	+29.6	+26.3	+35.8	+33.2	+40.4	+35.7
OpenVLA-OFT	Full-SFT	69.4±1.5	88.5±0.6	49.6±1.2	**66.3**±1.1	78.7±1.3	**70.3**±1.0	88.2±2.0	71.7±1.3
	One-SFT	12.8±0.8	49.6±1.9	23.0±1.0	30.0±2.2	23.3±2.3	34.5±0.3	42.0±0.1	29.3±1.5
	+ WCM (Ours)	**74.6**±3.6	**94.9**±0.1	**51.3**±0.4	65.8±2.2	**84.1**±0.6	63.6±1.4	**94.8**±1.6	**74.0**±1.8
	Δ	+61.8	+45.3	+28.3	+35.8	+60.8	+29.1	+52.8	+44.7

Baselines. For simulation, we adopt on-policy VLA-RL methods as baselines. To be specific, for π-style policies, we consider three recent VLA-RL baselines: Flow-Noise (Zhang et al., 2025), Flow-SDE (Ren et al., 2024), and π-stepNFT (Wang et al., 2026). For OpenVLA-OFT, we leverage the RLinf (Chen et al., 2025) implementations of standard PPO (Schulman et al., 2017) and GRPO (Liu et al., 2024) as baselines. We also compare WCM with history-exploiting baselines including frame stacking (Hausknecht & Stone, 2015), SPR (Schwarzer et al., 2020), recurrent architectures (Ni et al., 2021), and temporal models (Dosovitskiy et al., 2020). For real-robot, we adopt off-policy methods as baselines, specifically, AWR (Peng et al., 2019) for OpenVLA-OFT and RECAP (Intelligence et al., 2025a) for $\pi_{0.5}$. More details can be found in Appendix D.

Baseline results are taken from the corresponding papers. When training WCM, we follow the same model initialization, training budgets, optimization hyperparameters, and evaluation protocols as the corresponding baseline setup, while replacing the original critic with WCM.

4.2 Main Results

The main results are presented in Table 1 and Figure 3. Our method achieves strong performance across ManiSkill, MetaWorld and CALVIN benchmarks. Key findings are as follows:

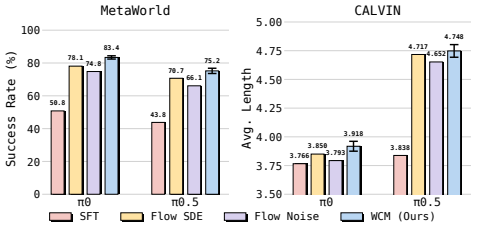


Figure 3: Results on MetaWorld and CALVIN with error bars (std).

(1) Effectiveness of WCM. Replacing the critics in Flow-SDE and PPO with WCM improves policy performance across ManiSkill, MetaWorld, and CALVIN under the evaluated settings. WCM achieves higher performance than the reported baseline results.

(2) Stable improvement from extremely low initial performance. When OpenVLA-OFT is initialized with no exposure to ManiSkill data and achieves 0.78% zero-shot success, WCM-based RL reaches 98.7%. In contrast, the previous critic struggles to learn under such extremely sparse initial rewards, as such sparse rewards lead to extremely sparse supervision. More details in Appendix D.4.

4.3 Generalization Performance

Generalization is a key evaluation criterion in VLA-RL. We evaluate our method under ManiSkill-OOD setting and LIBERO-Plus. Results are reported in Table 1 and Table 2. Key findings:

(1) OOD performance on ManiSkill. On ManiSkill, WCM achieves higher OOD success rates than the compared VLA-RL baselines. Notably, it also improves over π-StepNFT, a baseline specifically

Table 3: Detailed real-world experiment results. "+WCM" denotes using WCM as critic model. Δ denotes improvement from SFT. Results are taken from the first 50 trajectories after test starts.

Backbone	Method	Pick and Place			Deformable		Long-Horizon	Moving
		Carrot	Banana	Pepper	Cloth Folding	Towel Folding	Stovetop Cleaning	Conveyor Belt Sushi Picking
OpenVLA-OFT	SFT	24/50	11/50	19/50	15/50	16/50	1/50	9/50
	+ AWR	29/50	23/50	24/50	29/50	35/50	10/50	17/50
	+ WCM (Ours)	32/50	26/50	26/50	38/50	40/50	15/50	22/50
	Δ	8/50	15/50	7/50	23/50	24/50	14/50	13/50
$\pi_{0.5}$	SFT	25/50	31/50	34/50	21/50	24/50	4/50	13/50
	+ RECAP	33/50	37/50	40/50	32/50	33/50	27/50	18/50
	+ WCM (Ours)	44/50	38/50	43/50	38/50	35/50	33/50	24/50
	Δ	19/50	7/50	9/50	17/50	11/50	29/50	11/50

designed to mitigate critic overfitting and improve OOD performance, suggesting that the benefits of WCM extend to distribution-shifted settings.

(2) Comparison with Full-SFT on LIBERO-Plus. On LIBERO-Plus, starting from One-SFT, WCM reaches total success rates of 72.8%, 73.7%, and 74.0% for π_0, $\pi_{0.5}$, and OpenVLA-OFT, respectively, compared with 71.2%, 72.9%, and 71.7% for the corresponding Full-SFT models.

4.4 Real-World Performance

In the real-world setting, for better sample efficiency, we employ the off-policy pipeline. We select two RL algorithms as baselines: AWR for OpenVLA-OFT and RECAP for $\pi_{0.5}$, for which we use Gemma3-270M as the critic model. We evaluate all methods on 7 tasks: dynamic manipulation (rotating sushi picking), deformable object manipulation (cloth & towel folding), long-horizon (stovetop cleaning), and pick-and-place (carrot, pepper, banana) tasks. We train the SFT policy using 100 trajectories per task and perform 8 RL iterations, with 50 rollouts per task per iteration. All compared methods use matched initial demonstrations, rollout budgets, human-correction budgets, and evaluation trials; only the critic formulation is changed within each RL algorithm. Across the seven real-world tasks in Table 3, WCM with 25.1M trainable parameters yields higher observed success counts than the matched AWR or RECAP baselines with a VLM critic, under the same initial demonstrations, rollout budgets, human-correction budgets, and evaluation trials.

5 Analysis

5.1 Does Next-Latent Prediction Objective Help?

We test the hypothesis that history alone is not reliably sufficient for the tested value-only critics and that transition-aware supervision is helpful. The comparisons progressively vary temporal context, scalar value supervision, and predictive supervision under matched training settings, as shown in Figure 4. We evaluate three models (π_0, $\pi_{0.5}$, and OpenVLA-OFT) on ManiSkill and MetaWorld, extending the critic's input from single-frame to 5 frames. Key findings are as follows: **(1)** Adding history through frame stacking does not consistently improve performance and often degrades it as the history length increases. **(2)** The value-only Temporal Transformer does not show consistent gains from longer history **(3)** Incorporating a next-latent prediction objective enables better performance, and adding multi-frame history yields improvements over single-frame. **(4)** Since only a scalar target lacks supervision, an alternative approach is to adopt distributional RL, thereby turning regression into classification. We compare two representative methods (C51, HL-Gauss) and find them both underperforming WCM.

5.2 Can WCM Outperform History-Exploiting Baselines?

We apply several classic history-exploiting methods from generic RL to our VLA-RL setting, and study their performance as the history length varies. The conclusions are as follows:

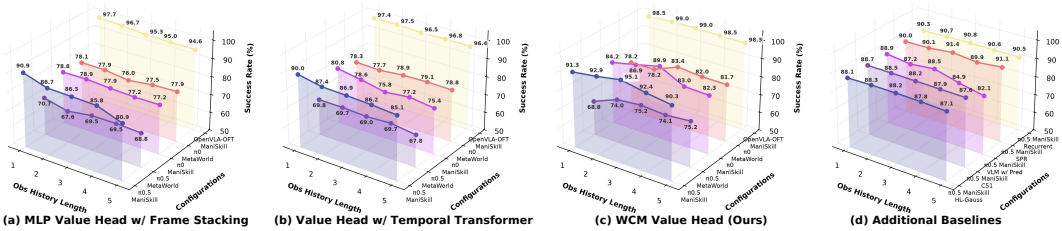


Figure 4: Ablation study on critic architectures and observation history lengths. Notably, Figure (b) also represents WCM without next-latent prediction ($\lambda = 0$).

(1) As shown in Figure 4(d), SPR and recurrent critics obtain modest and non-monotonic gains from additional history, and their best performances remain below that of WCM. (2) Figure 5 shows the training curves with history length 3 (across 3 seeds). WCM reaches higher success rate earlier in terms of RL training steps than the SPR and recurrent baselines. The recurrent baseline additionally requires BPTT, while SPR uses frame stacking and an EMA target encoder, which introduce different computational and optimization characteristics.

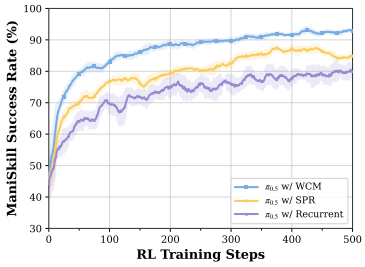


Figure 5: Training curves of WCM, SPR and recurrent method.

5.3 Beyond Success Rate: How Does Each Component Affect the Critic Itself?

We evaluate critic quality using quantitative critic-centric metrics, including value quality VOC-F1 (Zhai et al., 2025), prediction error (normalized L2), and probes for return, progress, state, action, and success prediction. For the component ablations, we keep the observation encoder, temporal trunk, value head, data, and optimization protocol fixed, changing only the indicated prediction objective or conditioning input. W/o SIGReg therefore tests the anti-collapse regularizer, w/o action-condition retains history and next-latent prediction but removes the executed action, and w/o prediction retains the history-conditioned value pathway but removes transition supervision. Results on all real-robot tasks with history length 3 are averaged over 3 seeds; metric definitions, per-task results, and error bars are provided in Appendix E. WCM obtains the highest value and probes scores and the lowest prediction error, while each ablation degrades critic quality. Moreover, inverting the motion direction in otherwise similar histories while preserving the current observation causes a sharp performance drop, supporting sensitivity to temporally ordered history rather than reliance on the current observation alone.

Table 4: Comparison on critic metrics.

Method	Value ↑	Pred. ↓	Probes ↑
WCM (Ours)	**0.97**	**0.12**	**0.64**
WCM w/o SIGReg	0.87	0.33	0.53
WCM w/o action-condition	0.91	0.25	0.49
WCM w/o pred.	0.90	N/A	0.53
WCM w/ freeze encoder	0.90	0.43	0.53
WCM w/ inverted dynamics	0.35	0.67	0.16
SPR Critic Model w/ pred.	0.94	0.21	0.60
VLM Critic Model w/ pred.	0.92	0.38	0.56

5.4 Is Longer Observation History Always Beneficial?

We conduct an ablation study by varying the observation history length of WCM from 1 to 5, and report the best IND performance in Figure 4(c). In our experiments, length 3 performed best on average. A possible explanation is that three consecutive frames provide richer short-term motion cues than one or two frames, while longer histories may introduce increasingly redundant information.

6 Conclusion

In this work, we studied history-conditioned critics for VLA-RL. In our experiments, adding observation history to a value-only critic was not reliably sufficient; action-conditioned next-latent prediction provided a complementary transition-aware signal that improved value learning and policy optimization. WCM couples these objectives in a shared language-conditioned temporal representation and can replace critics in both on-policy and off-policy pipelines. Across 149 simulated tasks and seven real-world tasks, WCM improves over the corresponding baseline critics, with improvement in several distribution-shifted and temporally ambiguous settings.

Ethics Statement

This work adheres to the ICLR Code of Ethics. Our study does not involve human subjects or animal experiments, and no personally identifiable or sensitive private information is used. All datasets and models are used in accordance with their respective licenses and usage guidelines. We have taken care to avoid potential privacy, safety, or discriminatory risks and are committed to maintaining transparency and integrity throughout the research process.

Reproducibility Statement

We make every effort to provide the details necessary to reproduce our experiments. The main paper and appendix describe the WCM architecture, training objectives, history configuration, optimization settings, reinforcement learning setup, and evaluation protocols. All experiments are conducted under consistent training and evaluation settings, and all simulation experiments are repeated with multiple random seeds. We further provide detailed definitions of the critic-centric metrics and frozen linear probes, together with per-task results and ablation studies in the appendix. For comparisons with prior methods, we follow their publicly available implementations or reported experimental settings and explicitly state the source of the reported results where applicable. We will release all the code, configurations, and evaluation scripts to facilitate further reproduction of our results.

References

Kartik Ahuja, Jason S Hartford, and Yoshua Bengio. Weakly supervised representation learning with sparse perturbations. *Advances in Neural Information Processing Systems*, 35:15516–15528, 2022.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

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.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Yonathan Efroni, Chi Jin, Akshay Krishnamurthy, and Sobhan Miryoosefi. Provable reinforcement learning with a short-term memory. In *International Conference on Machine Learning*, pp. 5832–5850. PMLR, 2022.

Senyu Fei, Siyin Wang, Li Ji, Ao Li, Shiduo Zhang, Liming Liu, Jinlong Hou, Jingjing Gong, Xianzhong Zhao, and Xipeng Qiu. Srpo: Self-referential policy optimization for vision-language-action models. *arXiv preprint arXiv:2511.15605*, 2025a.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025b.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International conference on machine learning*, pp. 2170–2179. PMLR, 2019.

Audrunas Gruslys, Rémi Munos, Ivo Danihelka, Marc Lanctot, and Alex Graves. Memory-efficient backpropagation through time. *Advances in neural information processing systems*, 29, 2016.

Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pp. 3875–3886. PMLR, 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Matthew J Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, pp. 141, 2015.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi_{0.6}$: A vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

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. *9th Annual Conference on Robot Learning*, 2025b.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: A steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2017.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

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.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Hamlet: Switch your vision-language-action model into a history-aware policy. *arXiv preprint arXiv:2510.00695*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Hao Li, Shuai Yang, Yilun Chen, Yang Tian, Xiaoda Yang, Xinyi Chen, Hanqing Wang, Tai Wang, Feng Zhao, Dahua Lin, et al. Cronusvla: Transferring latent motion across time for multi-frame prediction in manipulation. *arXiv e-prints*, pp. arXiv–2506, 2025a.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025b.

Michael Littman and Richard S Sutton. Predictive representations of state. *Advances in neural information processing systems*, 14, 2001.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025a.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*, 2025b.

Qinghua Liu, Alan Chung, Csaba Szepesvári, and Chi Jin. When is partially observable reinforcement learning not scary? In *Conference on Learning Theory*, pp. 5175–5220. PMLR, 2022.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger B Grosse. Reversible recurrent neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. *arXiv preprint arXiv:1709.04057*, 2017.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

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.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

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.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *International Conference on Learning Representations*, 2020.

Wenling Shang, Xiaofei Wang, Aravind Srinivas, Aravind Rajeswaran, Yang Gao, Pieter Abbeel, and Misha Laskin. Reinforcement learning with latent flow. *Advances in Neural Information Processing Systems*, 34:22171–22183, 2021.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

Satinder P Singh, Michael L Littman, Nicholas K Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 712–719, 2003.

Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12):1–83, 2022.

GigaBrain Team, Boyuan Wang, Bohan Li, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigabrain-0.5 m*: a vla that learns from world model-based reinforcement learning. *arXiv preprint arXiv:2602.12099*, 2026.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Siting Wang, Xiaofeng Wang, Zheng Zhu, Minnan Pei, Xinyu Cui, Cheng Deng, Jian Zhao, Guan Huang, Haifeng Zhang, and Jun Wang. π-stepnft: Wider space needs finer steps in online rl for flow-based vlas. *arXiv preprint arXiv:2603.02083*, 2026.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pp. 10674–10681, 2021.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, et al. Rlinf-vla: A unified and efficient framework for vla+ rl training. *arXiv preprint arXiv:2510.06710*, 2025.

Hongzhi Zang, Shu'ang Yu, Hao Lin, Tianxing Zhou, Zefang Huang, Zhen Guo, Xin Xu, Jiakai Zhou, Yuze Sheng, Shizhe Zhang, et al. Rlinf-user: A unified and extensible system for real-world online policy learning in embodied ai. *arXiv preprint arXiv:2602.07837*, 2026.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A How does the weight allocation of training objectives affect performance?

Another interesting question concerns the hyperparameter λ in our method, which balances the original value regression objective and the prediction objective. Specifically, $\lambda = 0$ uses only the value objective, while $\lambda = 1$ assigns equal weight to both. Results are shown in Figure 6. When the supervision signal is weak, even if IND performance remains reasonable, OOD performance is comparable to the original π_{RL} baseline. In contrast, with strong supervision, OOD performance remains competitive even if IND is slightly compromised.

(1) λ inherently controls a trade-off between next-latent prediction and value learning. Thus, neither extremely large nor small λ yields optimal results; there exists a stable intermediate range, with the best IND and OOD performance both occurring within $[0.3, 0.5]$.

(2) $\lambda = 0.9$ (dominated by the prediction objective) achieves better OOD performance than $\lambda = 0.1$ (dominated by the value regression objective), indicating that the next-latent prediction objective contributes positively to generalization.

(3) OOD performance is more sensitive to λ. Varying λ leads to a fluctuation of 10.6 percentage points in OOD, compared to only 2.7 percentage points in IND.

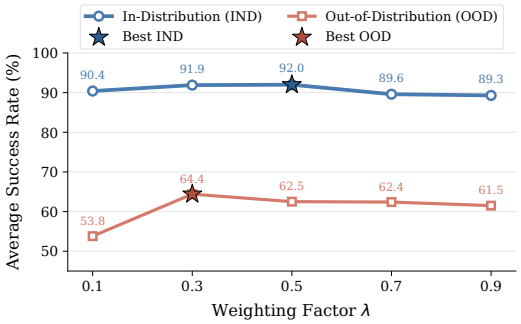


Figure 6: Effect of λ on IND and OOD performance. The plot shows the average success rate for both IND (blue) and OOD (red) across different values of λ, with stars indicating the best performance for each metric.

B Can the trained policy achieve sim-to-real transfer?

Since RL methods optimize for reward maximization, they are generally expected to achieve better generalization than SFT. To investigate this, we conduct two experiments, as shown in Table 5:

(1) We take (i) the SFT checkpoint trained on 16,800 simulated samples and (ii) the RL checkpoint obtained from 16,800 SFT samples followed by 285 RL steps (using an additional 18,240 simulated samples as batch size of a step is set to 64). Both models are deployed directly in the real world on a pick-and-place task, using similar camera view as in the simulation, and we compare their success rates. Among them, the SFT data in the simulation is generated by a rule-based planner, resulting in a form of data that supports parallel processing, low cost, and large-scale generation. In contrast, the rollout data is relatively more expensive to obtain.

We surprisingly find that the SFT model is completely unable to successfully grasp any object in the real world, despite our attempts with various different placement positions. In contrast, the policy obtained after RL achieves a high probability of successful grasping and placing in the central region of the camera's field of view, while exhibiting significant grasping deviations at the edges of the field of view.

(2) We start from a model fine-tuned with real-world SFT data (carrot pick-and-place) and continue RL training in simulation, examining whether this leads to improved success rates.

As the results shown, after fine-tuning with 50 real-world data samples, the policy achieves a success rate of only 6.9% in simulation. This result stands in stark contrast to the sim-to-real findings, where even training with 16,800 simulation samples fails to produce a policy capable of successfully performing the task in the real world.

We believe the above results are likely consistent with existing perspectives in the field regarding simulation data versus real-robot data: although simulation data is cheap and can be generated in large quantities, the simulation environment is completely idealized. Even if we introduce rule-based perturbations (such as changing backgrounds, objects, and positions, as done in our experiments), it remains difficult to obtain noise or disturbances similar to those in real environments (e.g., mo-

Table 5: Performance comparison under Sim IND, Sim OOD, and Real settings. The Sim OOD results are further broken down into vision, semantic, and execution dimensions. All methods are trained via RL on top of SFT initialization. Bold denotes the best performance.

Model	Method	Sim-IND	Sim-OOD				Real-Pick up object		
		avg.	vision	semantic	execution	avg.	carrot	banana	pepper
$\pi_{0.5}$	sim SFT	47.0	40.2	16.6	22.4	26.4	0/25	0/25	0/25
	+ sim RL	91.9	78.1	58.5	56.5	64.4	7/25	7/25	6/25
	Real SFT	6.9	5.1	5.9	5.2	5.4	13/25	2/25	4/25
	+ sim RL	73.5	42.1	35.9	35.8	37.9	11/25	8/25	9/25

tor temperature variations during inference), thereby making it difficult to provide corresponding robustness. Therefore, directly using simulation data for SFT may not yield significant OOD performance gains. This is reflected in our experiments, where simulation SFT failed to complete even a single pick-and-place task on the real robot, not even grasping succeeded. In contrast, using only 50 real-world data points enabled task completion in simulation. Moreover, since RL optimizes for reward maximization, it may offer better OOD performance compared to directly learning expert actions.

Furthermore, although we train the policy exclusively on the task of grasping carrot and placing it on plate, it still has a non-zero probability of successfully grasping bananas and peppers. After 1000 steps of reinforcement learning in simulation, the model's performance on grasping carrots slightly degrades, while its performance on grasping other objects significantly improves.

This is an interesting phenomenon. One possible explanation is that, in order to adapt to the simulation environment, the model adjusts its feature representation. This adjustment process may make the model more sensitive to features that are more common or easier to grasp in simulation, and these features happen to be more compatible with general-purpose grasping rather than with the in-distribution carrot in the simulation environment. This may also cause the model to partially forget its original ability to grasp carrots.

C Does Critic Model Affect Generalization?

We investigate the impact of the critic model on generalization performance. π-StepNFT highlights that PPO's scalar return-based critic is prone to overfitting. Based on this, we explore 2 questions:

(1) Whether OOD performance improves initially but deteriorates as the critic overfits. **(2)** What if we remove the critic in Flow-SDE by fixing the value function to $V(o) = 0$ for all observations, will this improves OOD performance?

We present the results in Figure 7. Key findings: **(1)** OOD performance of Flow-SDE shows a "dropping phenomenon", indicating overfitting in later stages. **(2)** Zero-value ablation achieves better OOD performance than Flow-SDE at comparable IND levels, suggesting overfitting may be linked to the critic model. **(3)** No overfitting is observed in WCM during the first 500 steps, though we do not claim it is entirely immune.

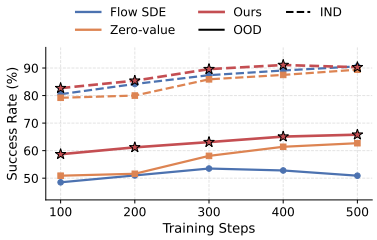


Figure 7: Training dynamics of Flow-SDE (dropping phenomenon) and Zero-Value.

D Details for On-policy and Off-policy Training

D.1 Algorithms

We present the detailed algorithms for on-policy and off-policy pipelines. The foundational algorithms we selected all require a critic model or a value function. Specifically, for on-policy learning,

we use PPO for the autoregressive model and Flow-SDE (an enhancement of PPO tailored for flow-matching models) for the flow-matching model. For off-policy, we use AWR for the AR model and RECAP for the flow-matching model. We leverage WCM to replace the original critic model or value function in the above algorithms, thus improving value estimation by adding historical observation. Below are the procedures for the training pipelines, where each method updates both the policy and WCM parameters through gradient-based optimization.

Algorithm 1: PPO with WCM Update

Input: Initial policy π_θ , initial WCM W_ϕ, language instruction ℓ.
Hyperparameters: loss weight λ , learning rate α, number of steps U , batch size B , observation history length K.
Output: Updated policy $\pi_{\theta'}$, updated WCM $W_{\phi'}$.

for *each optimization step* $u = 1, 2, \ldots, U$ **do**
  for *each episode* **do**
    Collect trajectory data $\{o_t, a_t, r_t, z_t, \hat{z}_{t+1}, \hat{V}_t, \ldots\}$ using π_θ and W_ϕ
  for *each batch of size* B **do**
    Compute the policy gradient loss:

$$\mathcal{L}_{\text{actor}} = -\mathbb{E}_t \left[\min \left(\rho_t(\pi) \hat{A}_t, \text{clip}(\rho_t(\pi), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right] \tag{11}$$

    where $\rho_t(\pi) = \frac{\pi(a_t|o_t)}{\pi_{\text{old}}(a_t|o_t)}$ and $\hat{A}_t$ is the estimated advantage using GAE. Compute the total critic loss from Equa.(6)-(10) using the stopped VLM backbone latent z_t^{sg}:

$$\mathcal{L}_{\text{critic}} = \mathcal{L}_{\text{value}}\big(W_\phi(z_t^{\text{sg}})\big) + \lambda\, \mathcal{L}_{\text{pred}}\big(W_\phi(z_t^{\text{sg}})\big) + \eta\, \mathcal{L}_{\text{SIGReg}}\big(W_\phi(z_t^{\text{sg}})\big) \tag{12}$$

    Update the critic and the actor separately:

$$\phi' = \phi - \alpha \nabla_\phi \mathcal{L}_{\text{critic}} \tag{13}$$

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}_{\text{actor}} \tag{14}$$

Algorithm 2: AWR with WCM Update

Input: Initial policy π_θ, WCM W_ϕ, Trajectory Buffer $\mathcal{B} = \varnothing$, language instruction ℓ.
Hyperparameters: loss weight λ, η, learning rate α, number of iterations I, batch size B, observation history length K
Output: Updated policy $\pi_{\theta'}$, updated WCM $W_{\phi'}$.

for *each episode* **do**
  Collect trajectory $\{o_t, a_t, r_t, z_t, \hat{z}_{t+1}, G_t, \ldots\}$ by teleop and add to $\mathcal{B}$
for *each iteration* $i = 1, 2, \ldots, I$ **do**
  for *each batch of size* B **do**
    Compute critic loss from Equa.(6)-(10) and update WCM via stochastic gradient descent:

$$\phi' = \phi - \alpha \nabla_\phi \mathcal{L} \tag{15}$$

  for *each batch of size* B **do**
    Compute the AWR loss for the policy update:

$$\mathcal{L}_{\text{actor}} = \mathbb{E}_t \left[-\log \pi(a_t|o_t) \exp\left(\frac{1}{\beta} \left(G_t - V(o_{t-K+1:t}) \right) \right) \right] \tag{16}$$

    Update policy via stochastic gradient descent:

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}_{\text{actor}} \tag{17}$$

  Collect trajectory data $\{o_t, a_t, r_t, z_t, \hat{z}_{t+1}, G_t, \ldots\}$ by policy rollout and add to $\mathcal{B}$.

Algorithm 3: RECAP with WCM Update

Input: Initial policy π_θ, WCM W_ϕ, Trajectory Buffer $\mathcal{B} = \varnothing$, language instruction ℓ.
Hyperparameters: loss weight λ, η, β, learning rate α, number of iterations I, batch size B, observation history length K, discount factor γ, advantage threshold ϵ_ℓ, TD target step N.
Output: Updated policy $\pi_{\theta'}$, updated WCM $V_{\phi'}$.

for *each episode* **do**
  Collect trajectory $\{o_t, a_t, r_t, z_t, \hat{z}_{t+1}, G_t, \dots\}$ by teleop and add to $\mathcal{B}$
for *each iteration* $i = 1, 2, \dots, I$ **do**
  for *each batch of size* B **do**
    Compute critic loss from Equa.(6)-(10) and update WCM via stochastic gradient descent:

$$\phi' = \phi - \alpha \nabla_\phi \mathcal{L} \tag{18}$$

  for *each batch of size* B **do**
    Estimate value and compute advantage:

$$A(o_t, a_t, \ell) = G_{t:t+N} + \gamma^N \hat{V}(o_{t+N-K+1:t+N}) - \hat{V}(o_{t-K+1:t}). \tag{19}$$

    Compute the RECAP loss for the policy update:

$$\mathcal{L}_{\text{actor}} = -\log \pi_\theta(a_t \mid o_t, \ell) - \beta \log \pi_\theta(a_t \mid \mathbf{1}(A(o_t, a_t, \ell) > \epsilon_\ell), o_t, \ell), \tag{20}$$

    Update policy via stochastic gradient descent:

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}_{\text{actor}} \tag{21}$$

  Collect trajectory data $\{o_t, a_t, r_t, z_t, \hat{z}_{t+1}, G_t, \dots\}$ by policy rollout and add to $\mathcal{B}$.

D.2 Training Details

On-policy training in simulation. Since the simulation environment enables low-cost rollout of large amounts of data, we adopt an on-policy method for training. For the π Intelligence et al. (2025a); Black et al. (2024) series models, we perform policy updates using Flow-SDE Chen et al. (2025) improved by WCM. Flow-SDE is a method proposed by the RLinf team, which combines DPPO Ren et al. (2024) and Flow-GRPO Liu et al. (2025a) to specifically adapt and enhance VLA models, offering strong engineering practicality and reproducibility. In the RLinf implementation, the critic model is a 3-layer MLP; they found that such a lightweight critic can fit task-specific requirements well. We replace it with our WCM, which takes the same input, i.e., the latent representation from the VLM backbone, passing through the WCM encoder, entering the temporal trunk, and outputs a scalar value. For OpenVLA-OFT Kim et al. (2025), we use the PPO implemented by RLinf Zang et al. (2025) and similarly replace the MLP-based critic with WCM.

On the ManiSkill Mu et al. (2021) benchmark, to ensure a fair comparison with baseline methods, we adopt exactly the same settings as RL4VLA Liu et al. (2025b), with 25 pick-and-place tasks and 2 major categories: in-distribution (IND) and out-of-distribution (OOD). IND includes various pick-and-place tasks with diverse objects, while OOD evaluates model performance under perturbations across three dimensions. We use the hyperparameters officially provided by RLinf without any additional tuning to ensure a perfectly fair comparison. Our batch size is set to 64, meaning that at each update step, the model observes 64 complete trajectories. The checkpoint used to initialize training is an SFT model trained on 16,800 trajectories generated by a rule-based planner. For OpenVLA-OFT, we also train from a checkpoint that is SFT only on LIBERO-Goal, Object, and Spatial, which achieves only 0.78% success rate. After about 600 steps of training, we are able to bring its performance close to near-perfect, with strong training stability.

We further evaluate on the MetaWorld Yu et al. (2020) and CALVIN Mees et al. (2022) benchmarks. While ManiSkill is limited to pick-and-place tasks, MetaWorld offers a broader variety of tasks, and CALVIN allows us to assess the model's long-horizon capabilities. Again, we follow the same hyperparameters used by RLinf for training and comparison. For MetaWorld, we report the final success rate; for CALVIN, we report the average task completion length, which is the official metric

recommended by CALVIN. This metric reflects how far the model can proceed before failing, with a minimum of 0 (unable to complete even one task) and a maximum of 5 (successfully completing all five tasks randomly selected by the benchmark). All simulation training results are averaged over three runs with different seeds, reporting the error bars (std).

Off-policy training in real-world. We have also conducted extensive work on real-robot RL to demonstrate the effectiveness of our method in physical environments. Since real-world data is difficult to acquire and rollouts risk damaging the hardware or motors of the manipulator (We set hardware limits to prevent such incidents), on-policy training is less practical in our setting. Instead, we adopt two off-policy RL algorithms for real-robot training: AWR Peng et al. (2019) and RECAP Intelligence et al. (2025a). The former is a classic offline RL method; however, in our implementation, we do not use an additional transition model, but simply supplement the data buffer with online rollout data, and thus we prefer to call it off-policy rather than offline. RECAP was also used in $\pi^*_{0.6}$ Intelligence et al. (2025a), where it achieved substantial performance improvements. We use AWR to update the OpenVLA-OFT model and RECAP to update the $\pi_{0.5}$ model. It is worth noting that in our implementation, the pipelines of the two methods are largely identical, with the only difference being the loss function used for policy update.

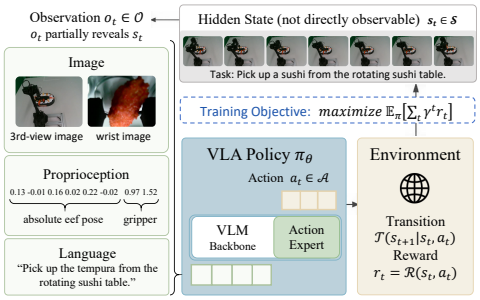


Figure 8: Illustration of the rollout process. Starting from an observation o_t partially revealing hidden state s_t, the policy π_θ generate action a_t and the environment provides the transition and reward.

All our real-robot experiments are conducted on the WidowX-250S robot arm, using both third- and wrist-camera views, as shown in Figure 9. We employ delta end-effector pose as the control method and use absolute end-effector pose along with a two-finger gripper state (representing openness) as proprioceptive input. We built a master-slave teleoperation data collection pipeline based on ROS2 and collected an initial set of 100 trajectories for each task. We trained our WCM on seven tasks separately, providing a total of approximately 250k transition pairs. For the policy, three pick-and-place tasks share a single SFT policy, two deformable object manipulation tasks share another SFT policy, and the remaining two tasks are each SFT separately. In each round of RL update, we collect all successful rollouts. For unsuccessful rollouts, we apply human-in-the-loop corrections to turn them into successful trajectories (for reward, we assign a reward of 1 at the success point). Baseline methods also receive matched human feedback budgets, and all methods are evaluated entirely without human intervention during testing. In each RL update round, 50 trajectories (including both failed and successful ones) are collected per task, and we use new data from each task to update the WCM. Each reported data point is obtained after 8 iterations of RL updates.

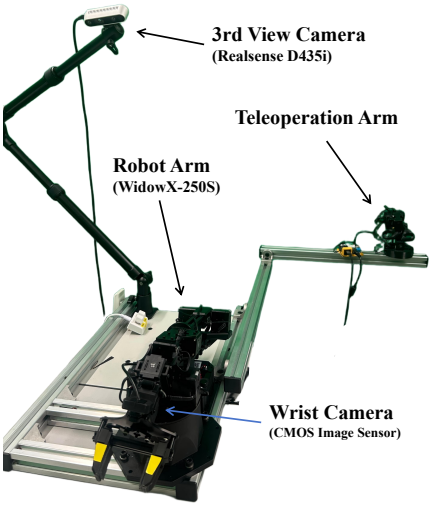


Figure 9: Demonstration of the Real-Robot Experiment Environment.

In our implementations of AWR and RECAP, we selected a value model comprising a SigLip 400M Encoder and a Gemma 270M Backbone guided by the reference $\pi^*_{0.6}$ Intelligence et al. (2025a), while adopting a training paradigm identical to that of our World Critic Model (+WCM). As demonstrated in Table 3, higher observed success counts for WCM than the corresponding baseline critic across the evaluated tasks. These results are consistent with two design differences of WCM: first, unlike conventional scalar regression, our WCM incorporates next-latent prediction objectives, fostering a representation that preserves value-relevant temporal context and action-conditioned transition cues relevant to manipulation progress; second, by integrating historical observations, our

critic can exploit ordered observation history, which helps distinguish visually similar interaction stages. Empirically, these advantages manifest in distinct behavioral insights across two challenging setups. In deformable object manipulation, single-frame critic models trained via reinforcement learning tend to drive the end-effector into the tabletop, causing motor stalls; this occurs because reaching the tabletop's z-coordinate correlates with successful grasping, yet lacking historical context, the single-frame model cannot discern whether this physical obstruction is beneficial or detrimental from a static snapshot. Conversely, in the visualized trajectories, the WCM-trained policy exhibits fewer tabletop collisions, while frames involving such collisions are assigned noticeably lower values. Furthermore, in the Conveyor Belt Sushi Picking task, the single-frame critic baseline struggles to improve due to a pronounced post-grasp latency of several seconds that leads to collisions with adjacent objects or items being dragged away; in contrast, the WCM-based variant achieves a higher observed success rate and exhibits fewer post-grasp execution failures.

Computational resources and settings. For simulation, all experimental data points were obtained on an 8 $\times$ H100 machine, including training and evaluation. For real-robot experiments, all training was conducted on 8 $\times$ H100 GPUs, while inference was performed on a local workstation with RTX5090. The control frequency for all real-robot experiments was 10 Hz, and the action chunk size was set to 5. Thus, each action chunk spans 0.5 seconds, with a new observation being received at each control step (every 0.5 s).

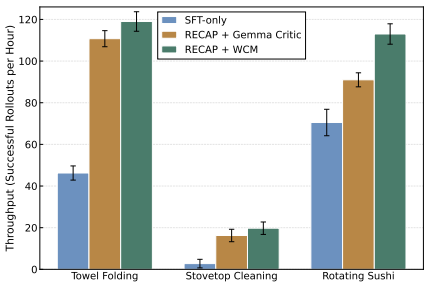


Figure 10: Throughput comparison of different policy variants across three tasks. Throughput is measured as the number of successful rollouts per hour. Bars represent mean values over four evaluation runs, with error bars indicating standard deviation. The SFT-only model has lower throughput across all three tasks, while both RECAP-trained variants achieve higher throughput. Among the two RECAP variants, the WCM-based critic consistently outperforms the original Gemma-based critic.

Gradient Flow and Prediction Targets. The current latent z_t and next-step target latent z_{t+1} are produced by the same trainable WCM observation encoder. We do not apply stop-gradient to either branch within WCM: the prediction loss is back-propagated through both the predicted branch and the target latent branch, such that the observation encoder is jointly updated by the prediction objective. SIGReg regularizes the jointly learned latent space and prevents representational collapse, allowing us to optimize the source and target representations without introducing a separate EMA target encoder. When WCM takes latent features extracted from the policy VLM backbone as observations, these VLM features are detached before being passed to WCM. Therefore, gradients from the WCM objectives update only WCM and do not propagate into the policy VLM backbone.

D.3 Task-level Throughput

We evaluated the task-level throughput of the $\pi_{0.5}$ model across three tasks: towel folding, stovetop cleaning, and rotating sushi picking. Throughput is measured as the number of successful rollouts per hour. The results are presented in Figure 10.

The SFT-only model yields the lowest throughput, due to its low initial success rate and inefficient action execution, characterized by frequent pauses and small-magnitude movements, which prolong each rollout. After RL fine-tuning, successful-rollout throughput is higher across all three tasks.

Among the two RECAP-trained variants, the model using the WCM-based critic consistently outperforms the Gemma VLM-based critic variant across all tasks, showing that the WCM-based variant achieves higher successful-rollout throughput than the Gemma-critic variant across the three evaluated tasks.

D.4 Zero-Shot Learning under Extremely Sparse Initial Rewards

We further investigate whether WCM remains effective when reinforcement learning starts from an extremely challenging initialization with almost no successful experience.

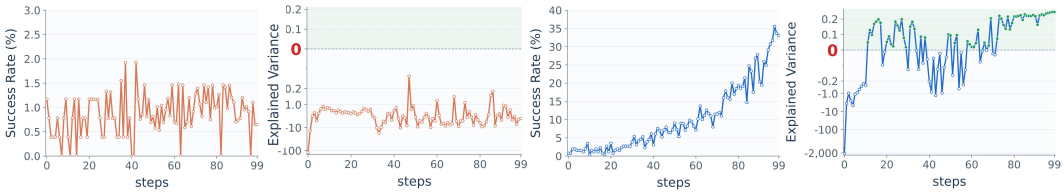


Figure 11: Zero-shot RL training under extremely sparse initial rewards. The original critic receives limited learning signals and exhibits unstable value learning, while WCM provides additional transition-aware supervision through next-latent prediction, resulting in more stable critic optimization.

We consider the zero-shot OpenVLA-OFT setting, where the policy is initialized from a LIBERO-Goal checkpoint without prior interaction with the target environment. Under this setting, the initial policy achieves only 0.78% success rate, resulting in extremely sparse positive reward signals.

Such sparse feedback creates a challenge for critic-based reinforcement learning. Before meaningful successful trajectories are collected, the critic receives limited supervision for distinguishing useful intermediate states from unsuccessful behaviors. Consequently, value estimation can become unstable and provide weak learning signals for policy improvement.

Unlike a value-only critic, WCM introduces an additional transition-aware prediction objective during critic learning. Specifically, WCM learns a shared temporal representation optimized jointly by return prediction and action-conditioned next-latent prediction.

Importantly, the predicted latent is only used as an auxiliary training signal for representation learning and is not provided as an additional input during policy inference.

Figure 11 compares PPO training with the original critic and PPO training with WCM. The original critic struggles to obtain stable improvement during early training, where the explained variance remains low due to the lack of informative reward signals. In contrast, WCM provides additional transition-aware supervision for learning the critic representation, leading to more stable critic optimization and subsequent policy improvement.

After reinforcement learning fine-tuning, WCM reaches a success rate of 98.7%, compared with the 0.78% zero-shot initialization. These results indicate that transition-aware auxiliary supervision can improve critic learning stability in settings where reward-based supervision is initially extremely limited.

D.5 Training Curve

To better understand the optimization behavior and convergence of different configurations, we plot the training curves of four settings in Figure 12. As shown in the figure, these settings exhibit stable improvement over training steps, though their convergence speeds and final performance vary across configurations.

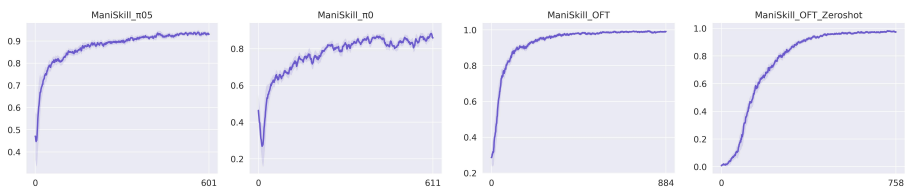


Figure 12: Training curves across four different settings.

D.6 Confidence Intervals for Real-World Evaluation

To quantify the uncertainty of the real-world success rates reported in Table 3, we additionally compute 95% Wilson score confidence intervals for each post-RL method on every task. Each

Table 6: Success rates (%) with 95% Wilson score confidence intervals on the seven real-world tasks. Each entry is reported as success rate [lower, upper], based on 50 evaluation trials per task.

Backbone	Method	Carrot	Banana	Pepper	Cloth Folding	Towel Folding	Stovetop Cleaning	Conveyor Belt Sushi Picking
OpenVLA-OFT	+AWR	58.0 [44.2, 70.6]	46.0 [33.0, 59.6]	48.0 [34.8, 61.5]	58.0 [44.2, 70.6]	70.0 [56.2, 80.9]	20.0 [11.2, 33.0]	34.0 [22.4, 47.8]
OpenVLA-OFT	+WCM (Ours)	64.0 [50.1, 75.9]	52.0 [38.5, 65.2]	52.0 [38.5, 65.2]	76.0 [62.6, 85.7]	80.0 [67.0, 88.8]	30.0 [19.1, 43.8]	44.0 [31.2, 57.7]
$\pi_{0.5}$	+RECAP	66.0 [52.2, 77.6]	74.0 [60.4, 84.1]	80.0 [67.0, 88.8]	64.0 [50.1, 75.9]	66.0 [52.2, 77.6]	54.0 [40.4, 67.0]	36.0 [24.1, 49.9]
$\pi_{0.5}$	+WCM (Ours)	88.0 [76.2, 94.4]	76.0 [62.6, 85.7]	86.0 [73.8, 93.0]	76.0 [62.6, 85.7]	70.0 [56.2, 80.9]	66.0 [52.2, 77.6]	48.0 [34.8, 61.5]

evaluation consists of $n = 50$ independent trials, with success or failure treated as a Bernoulli outcome. For an observed success rate $\hat{p} = k/n$, the Wilson interval is computed as

$$\mathrm{CI}_{95\%} = \frac{\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}, \tag{22}$$

where $z = 1.96$ for a 95% confidence level. Table 6 reports the resulting intervals for all post-RL comparisons. These intervals provide an estimate of evaluation uncertainty due to the finite number of real-world trials.

E Critic Metrics

Beyond evaluating downstream policy performance, we further analyze the quality of the learned critic itself. Specifically, we evaluate WCM from three complementary perspectives: (1) value estimation quality, (2) transition prediction accuracy, and (3) information encoded in the learned latent representation.

All critic metrics are computed with a frozen WCM checkpoint. For representation analysis, we extract latent features from the trained critic and fit lightweight linear probes only on the training split, while reporting results on a disjoint validation split. Therefore, probe objectives do not affect WCM optimization and only measure the information already encoded in the learned representation.

E.1 Value Quality

We evaluate whether the learned value function preserves the temporal ordering of task progress using VOC-F1, following the value-ordering evaluation introduced in VLAC and adapted to our state-value critic setting.

Given an episode trajectory with predicted values $\{V_t\}_{t=1}^{T}$, we compute the value ordering correlation as

$$\mathrm{VOC} = \rho_{\mathrm{Spearman}}(\{V_t\}_{t=1}^{T}, \{t\}_{t=1}^{T}), \tag{23}$$

where ρ_{Spearman} denotes the Spearman rank correlation.

To measure whether the critic assigns consistent values under reversed temporal contexts, we additionally compute the reversed ordering score (VROC) by feeding the reversed observation history into WCM.

The final value-ordering metric is computed as

$$\text{VOC-F1} = \frac{2 \cdot \mathrm{VOC} \cdot \mathrm{VROC}}{\mathrm{VOC} + \mathrm{VROC}}. \tag{24}$$

We report the macro-average over successful episodes. Note that VROC is used as a critic-compatible diagnostic metric and does not exactly reproduce the pairwise comparison protocol of VLAC, since WCM predicts state values rather than pairwise progress scores.

E.2 Next-Latent Prediction Quality

WCM contains an action-conditioned prediction head that predicts the latent representation of the next observation. We evaluate this auxiliary objective independently from downstream policy performance.

Given the predicted next latent $\hat{z}_{t+1}$ and the encoded next latent z_{t+1}, we compute the normalized latent prediction error:

$$\text{PredErr} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left\|\hat{z}_{t+1}^{(i)} - z_{t+1}^{(i)}\right\|_2}{\left\|z_{t+1}^{(i)}\right\|_2}. \tag{25}$$

Lower values indicate more accurate next-latent prediction.

To verify that the predictor captures transition-related information rather than simply copying the current representation, we additionally compare against two non-learning baselines:

- **Persistence**: directly uses the current latent as prediction, i.e., $\hat{z}_{t+1} = z_t$.
- **Zero-action**: replaces the executed action with a zero vector before applying the prediction head.

E.3 Latent Representation Probes

To investigate what information is encoded in WCM representations, we perform post-hoc linear probing on frozen latent features. We evaluate three learned representations:

- observed-state latent h_t,
- context latent from the temporal trunk,
- predicted next latent $\hat{z}_{t+1}$.

The ground-truth next latent z_{t+1} obtained from the next observation is additionally reported as an upper-bound reference. Since it is not generated by WCM, it is excluded from the main probe score. The probe targets cover five aspects of task-relevant information:

- future return,
- normalized episode progress,
- proprioceptive state,
- executed action,
- episode success.

For continuous targets, including return, progress, state, and action, we fit a ridge regression probe:

$$\hat{y} = Wh + b. \tag{26}$$

We normalize the probe performance using the improvement over a mean predictor:

$$S_{\text{reg}} = \text{clip}\left(\frac{\text{MSE}_{\text{base}} - \text{MSE}_{\text{probe}}}{\text{MSE}_{\text{base}}}, 0, 1\right). \tag{27}$$

For binary success prediction, we train a logistic regression probe and normalize the ROC-AUC improvement over random guessing:

$$S_{\text{cls}} = \text{clip}\left(\frac{\text{AUC}_{\text{probe}} - 0.5}{1 - 0.5}, 0, 1\right). \tag{28}$$

Finally, the overall probe score is computed as the macro-average over all latent sources and prediction targets:

$$S_{\text{probe}} = \frac{1}{|\mathcal{H}||\mathcal{Y}|}\sum_{h\in\mathcal{H}}\sum_{y\in\mathcal{Y}} S(h, y). \tag{29}$$

E.4 Per-task Critic Metric Details

To provide a more fine-grained analysis of critic quality, we report the per-task breakdown of all critic-centric metrics on the seven real-world manipulation tasks. The table organizes the evaluation into three dimensions: value quality, transition prediction quality, and latent representation quality.

The representation quality is further decomposed into five frozen linear probes measuring encoded information about return, episode progress, proprioceptive state, executed action, and task success.

To compute the overall score, all metrics are first normalized into a unified higher-is-better scale. Metrics with higher values preferred (VOC-F1 and probe scores) directly use their normalized values, while lower-is-better metrics (prediction error) are inverted. The reported average is the mean over all normalized metric dimensions.

What do the critic-centric metrics reveal? Beyond the aggregate improvement of WCM, the per-task results reveal several interesting patterns about what different components contribute to the learned critic representation.

Sensitivity to transition structure. The most pronounced behavior appears under the inverted-dynamics setting. When the temporal motion direction is inverted while the task, scene content, and overall observation appearance remain otherwise matched, critic quality degrades substantially across all metrics. In particular, VOC-F1 drops from 0.97 to 0.35, while the normalized prediction error increases from 0.12 to 0.67. The degradation is similarly strong for the frozen probes, with Return, Progress, State, Action, and Success decreasing from $0.63/0.80/0.49/0.32/0.94$ to $0.15/0.28/0.10/0.05/0.23$, respectively. Importantly, this intervention does not introduce an obvious appearance-level OOD cue: it preserves the task and visual content and primarily breaks the temporal consistency between consecutive observations and actions. Therefore, the sharp degradation is difficult to explain purely by visual distribution shift. Instead, it suggests that the critic representation is sensitive to whether the observed history follows a coherent transition structure. This result provides direct evidence that the benefit of temporal context does not arise merely from exposing the critic to additional observations; the ordering and action-dependent transition relations within the history are themselves important.

SIGReg mainly preserves fine-grained structure rather than only coarse success information. Removing SIGReg consistently degrades critic quality: VOC-F1 decreases from 0.97 to 0.87, the normalized prediction error increases from 0.12 to 0.33, and the Return Probe drops from 0.63 to 0.48. Interestingly, the Success Probe remains relatively high at 0.84, despite the much larger degradation in value ordering and transition prediction. This discrepancy suggests that coarse outcome information can remain linearly decodable even when the representation is no longer sufficiently structured for fine-grained value estimation or transition prediction. In other words, distinguishing whether a trajectory is likely to succeed or fail may require substantially less representational structure than correctly ordering intermediate states according to value or modeling how those states evolve. From this perspective, SIGReg appears to contribute not merely by preserving binary success-related information, but by maintaining a richer and more fine-grained organization of the critic representation that is useful for both value learning and temporal prediction.

Action conditioning is particularly important for action-relevant representations. Removing action conditioning produces a different failure pattern. The resulting critic still retains moderate value and prediction quality, achieving a VOC-F1 of 0.91 and a normalized prediction error of 0.25. However, the Action Probe decreases substantially from 0.32 to 0.18, while the Return Probe also decreases from 0.63 to 0.50. This suggests that next-latent prediction remains partially feasible even without explicitly conditioning on the action, since consecutive observations still contain substantial temporal correlation. However, without the action variable, the learned temporal representation contains considerably less information about which transition is induced by which control command. Therefore, action conditioning does more than improve prediction accuracy: it encourages the critic representation to encode action-dependent transition information that is directly relevant to control.

The benefit of temporal modeling varies across task types. We also observe a consistent task-dependent pattern across several metrics. The three pick-and-place tasks (Carrot, Banana, and Pepper) tend to obtain slightly lower VOC-F1 and probe scores, together with higher prediction error, than Cloth Folding, Towel Folding, Stovetop Cleaning, and Conveyor Belt Sushi Picking. For example, under WCM, the prediction errors for Carrot/Banana/Pepper are $0.14/0.17/0.13$, compared with $0.08/0.09/0.10/0.10$ for Cloth/Towel/Stovetop/Sushi.

One possible interpretation is that the latter group contains more extended temporal structure, where the current observation alone may be less sufficient to determine task progress or the appropriate future behavior. In such cases, history can provide additional information about how the current

state was reached and which stage of the manipulation is being executed. By contrast, pick-and-place tasks may depend more strongly on relatively local geometric information, such as the current object and end-effector configuration, making additional temporal context comparatively less informative. We emphasize that this task-level comparison should be interpreted as an empirical trend rather than a controlled test of task horizon. Nevertheless, the observation is consistent with the hypothesis that temporal critic representations become increasingly useful when successful control depends on reasoning over a longer or more structured sequence of state transitions.

Taken together, these results suggest that the improvement of WCM cannot be attributed to simply increasing critic capacity or exposing the critic to more observations. The inverted-dynamics experiment shows that the *correct temporal structure* of the history matters; the SIGReg ablation shows that preserving a well-structured representation is important for fine-grained value and transition information; and the action-conditioning ablation shows that explicit control information is necessary for learning action-relevant transition representations. These observations collectively support our main hypothesis that history becomes useful to the critic when it is accompanied by transition-aware supervision that constrains how temporal information should be represented.

F Value Curve Visualization of WCM

To better illustrate WCM's contribution to the RL process, we visualize the value curves of successful and failed trajectories in both simulation and real-world tasks.

For simulation, we construct 5k successful trajectories and 2k failed ones. Among the failed trajectories, 1k are random end-effector (EEF) movements (translation + rotation), and the remaining 1k are "near-success" trajectories where the planner is perturbed (e.g., adding a coordinate offset to the target grasping position), causing task failure. We train a single WCM on all 7k trajectories and evaluate it on unseen trajectories. The results are shown in Figure 13(d)–(f). For successful trajectories, the value curve increases monotonically, which is expected given that the trajectories in simulation are idealized and near-optimal. The two failure types exhibit distinct patterns. For random EEF movement, which shows no tendency toward task completion, the value depends primarily on the EEF's position during random wandering; translation causes only slight value degradation as the EEF may occasionally move closer to or farther from the target, while rotation leads to a sharp value drop, as it deviates completely from successful behavior. For near-success trajectories, the value first rises and then falls, mirroring the trajectory's initial progression toward success followed by eventual failure.

For real-world tasks, we train WCM with 500 trajectories per task (including both successes and failures). The results are shown in Figure 13(a)–(c). Due to the less idealized nature of the real-world environment and the inevitable inclusion of suboptimal actions in the collected trajectories, even successful trajectories do not exhibit strictly monotonic value curves. Value stagnation or decline is primarily caused by factors such as pauses during grasping, collisions with the table or objects, unfavorable grasp poses, or imperfect task completion (e.g., misplaced items, misaligned cloth folding). For failed trajectories, the value curves generally exhibit a downward trend.

G Implementation Details and Hyperparameters

WCM training. The temporal representation has a latent dimension of 384 and is processed by a 6-layer Transformer with 8 attention heads. The dynamics predictor contains 3 layers. We use a pretrained ViT-S/16 as the visual encoder and CLIP ViT-B/32 for language encoding; the visual encoder is trainable while the language encoder is frozen.

We set the next-latent prediction coefficient λ in the range of $[0.3, 0.5]$ across different experimental settings, while the SIGReg coefficient is set to 0.01. We optimize WCM with a learning rate of 5×10^{-5}, weight decay of 10^{-3}, and Adam coefficients $(\beta_1, \beta_2) = (0.9, 0.95)$. The default configuration uses a per-device batch size of 64, a gradient clipping norm of 1.0, and BF16 precision.

Reinforcement learning. RL fine-tuning is implemented within the RLinf framework. WCM receives the most recent three latent observations during RL training, consistent with the default history setting used above. The WCM/value component is optimized with a learning rate of 5×10^{-5}.

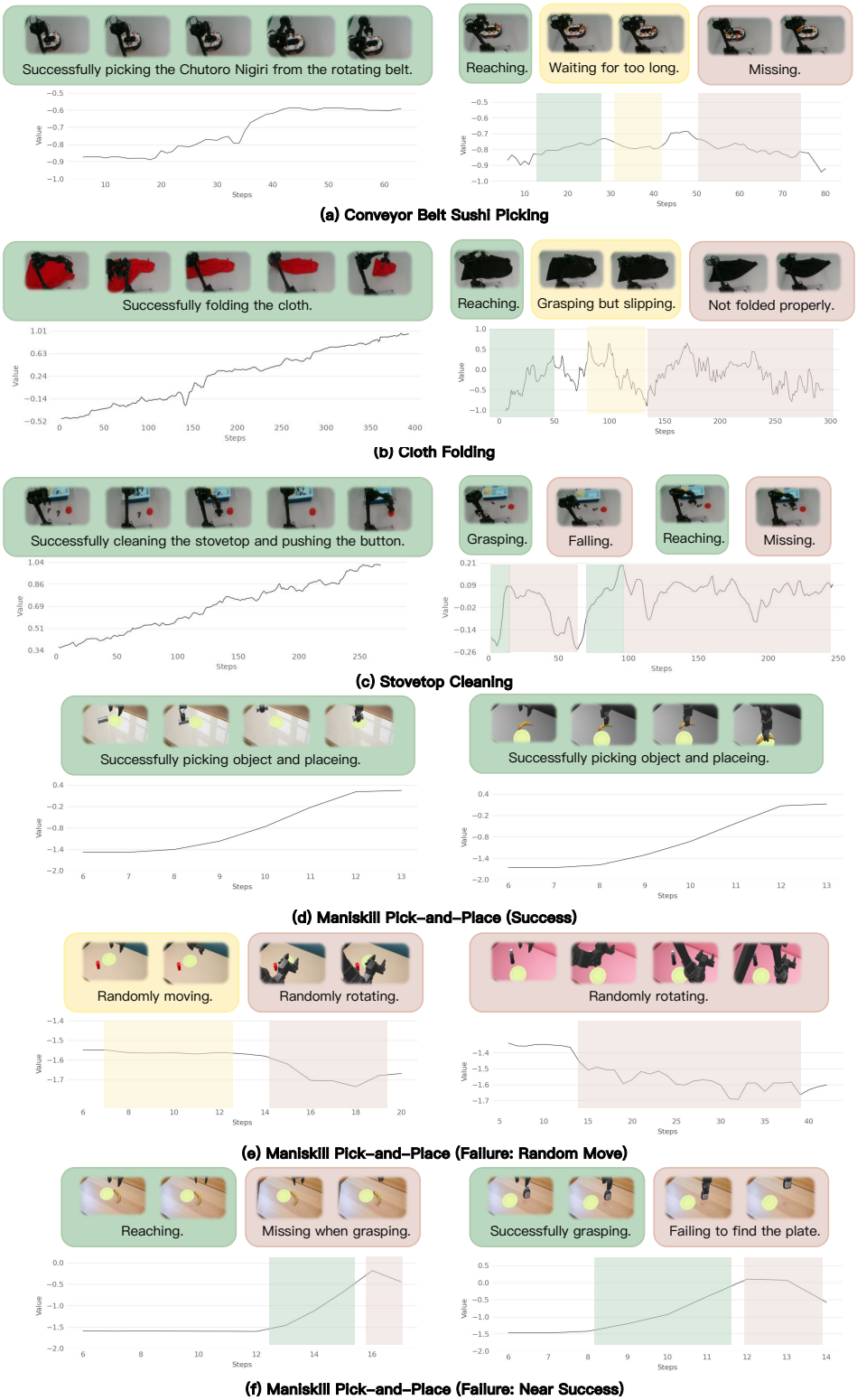


Figure 13: WCM value curve visualization across 6 tasks (3 real-world + 3 simulation).

Table 7: Per-task critic metric breakdown. VOC-F1 evaluates value ordering quality, normalized L2 evaluates next-latent prediction accuracy, and probe metrics measure task-relevant information encoded in the learned representation. Results are reported as mean standard deviation across three training runs with different seeds.

Method	Task	Value	Prediction	Frozen Linear Probe				
		VOC-F1 ↑	Norm. L2 ↓	Return ↑	Progress ↑	State ↑	Action ↑	Success ↑
WCM (Ours)	Carrot	0.94 ± 0.04	0.14 ± 0.03	0.61 ± 0.04	0.79 ± 0.05	0.46 ± 0.03	0.30 ± 0.05	0.93 ± 0.04
	Banana	0.93 ± 0.02	0.17 ± 0.04	0.60 ± 0.02	0.78 ± 0.02	0.48 ± 0.03	0.30 ± 0.01	0.91 ± 0.01
	Pepper	0.96 ± 0.02	0.13 ± 0.01	0.61 ± 0.03	0.77 ± 0.02	0.47 ± 0.01	0.31 ± 0.05	0.92 ± 0.03
	Cloth	0.99 ± 0.01	0.08 ± 0.01	0.64 ± 0.03	0.82 ± 0.02	0.49 ± 0.03	0.35 ± 0.04	0.95 ± 0.01
	Towel	0.99 ± 0.01	0.09 ± 0.02	0.63 ± 0.03	0.81 ± 0.02	0.51 ± 0.06	0.32 ± 0.03	0.96 ± 0.01
	Stovetop	0.98 ± 0.01	0.10 ± 0.03	0.66 ± 0.04	0.80 ± 0.01	0.50 ± 0.01	0.33 ± 0.03	0.94 ± 0.01
	Sushi	0.98 ± 0.01	0.10 ± 0.03	0.65 ± 0.04	0.83 ± 0.03	0.51 ± 0.03	0.32 ± 0.03	0.96 ± 0.01
	Average	0.97 ± 0.01	0.12 ± 0.01	0.63 ± 0.01	0.80 ± 0.01	0.49 ± 0.01	0.32 ± 0.01	0.94 ± 0.01
WCM w/o SIGReg	Carrot	0.85 ± 0.02	0.35 ± 0.04	0.46 ± 0.05	0.68 ± 0.04	0.36 ± 0.02	0.22 ± 0.03	0.81 ± 0.04
	Banana	0.86 ± 0.05	0.36 ± 0.02	0.47 ± 0.03	0.69 ± 0.01	0.36 ± 0.01	0.21 ± 0.04	0.83 ± 0.05
	Pepper	0.86 ± 0.03	0.34 ± 0.05	0.45 ± 0.04	0.70 ± 0.02	0.35 ± 0.04	0.23 ± 0.05	0.83 ± 0.03
	Cloth	0.88 ± 0.03	0.32 ± 0.01	0.48 ± 0.04	0.73 ± 0.05	0.40 ± 0.03	0.25 ± 0.03	0.84 ± 0.01
	Towel	0.89 ± 0.03	0.31 ± 0.04	0.49 ± 0.05	0.71 ± 0.02	0.39 ± 0.02	0.24 ± 0.02	0.86 ± 0.04
	Stovetop	0.88 ± 0.04	0.31 ± 0.02	0.51 ± 0.04	0.72 ± 0.03	0.37 ± 0.05	0.26 ± 0.02	0.87 ± 0.02
	Sushi	0.87 ± 0.02	0.32 ± 0.01	0.50 ± 0.03	0.74 ± 0.03	0.37 ± 0.03	0.27 ± 0.03	0.85 ± 0.01
	Average	0.87 ± 0.01	0.33 ± 0.01	0.48 ± 0.02	0.71 ± 0.01	0.37 ± 0.01	0.24 ± 0.01	0.84 ± 0.01
WCM w/o action-condition	Carrot	0.89 ± 0.02	0.28 ± 0.02	0.48 ± 0.02	0.65 ± 0.03	0.30 ± 0.04	0.16 ± 0.03	0.75 ± 0.06
	Banana	0.87 ± 0.03	0.28 ± 0.03	0.48 ± 0.01	0.63 ± 0.03	0.31 ± 0.03	0.17 ± 0.07	0.74 ± 0.03
	Pepper	0.90 ± 0.02	0.25 ± 0.03	0.48 ± 0.01	0.65 ± 0.05	0.31 ± 0.04	0.17 ± 0.04	0.73 ± 0.02
	Cloth	0.91 ± 0.05	0.26 ± 0.01	0.51 ± 0.04	0.68 ± 0.06	0.33 ± 0.01	0.21 ± 0.02	0.77 ± 0.03
	Towel	0.93 ± 0.02	0.22 ± 0.01	0.51 ± 0.03	0.66 ± 0.01	0.34 ± 0.04	0.18 ± 0.02	0.78 ± 0.05
	Stovetop	0.93 ± 0.02	0.23 ± 0.02	0.50 ± 0.04	0.69 ± 0.01	0.35 ± 0.03	0.20 ± 0.01	0.79 ± 0.05
	Sushi	0.92 ± 0.03	0.24 ± 0.03	0.53 ± 0.04	0.67 ± 0.01	0.35 ± 0.01	0.18 ± 0.04	0.76 ± 0.01
	Average	0.91 ± 0.01	0.25 ± 0.01	0.50 ± 0.01	0.66 ± 0.01	0.33 ± 0.01	0.18 ± 0.01	0.76 ± 0.01
WCM w/o Pred.	Carrot	0.88 ± 0.03	N/A	0.48 ± 0.03	0.69 ± 0.04	0.40 ± 0.01	0.19 ± 0.04	0.77 ± 0.05
	Banana	0.87 ± 0.01	N/A	0.48 ± 0.04	0.69 ± 0.04	0.40 ± 0.04	0.21 ± 0.06	0.76 ± 0.03
	Pepper	0.89 ± 0.02	N/A	0.50 ± 0.01	0.70 ± 0.03	0.38 ± 0.03	0.19 ± 0.04	0.78 ± 0.03
	Cloth	0.91 ± 0.04	N/A	0.51 ± 0.03	0.73 ± 0.06	0.44 ± 0.03	0.22 ± 0.07	0.80 ± 0.04
	Towel	0.92 ± 0.02	N/A	0.52 ± 0.01	0.71 ± 0.04	0.42 ± 0.03	0.24 ± 0.04	0.79 ± 0.04
	Stovetop	0.92 ± 0.03	N/A	0.53 ± 0.02	0.72 ± 0.01	0.41 ± 0.02	0.23 ± 0.04	0.82 ± 0.03
	Sushi	0.91 ± 0.01	N/A	0.53 ± 0.03	0.74 ± 0.04	0.43 ± 0.02	0.25 ± 0.03	0.81 ± 0.03
	Average	0.90 ± 0.01	N/A	0.51 ± 0.01	0.71 ± 0.02	0.41 ± 0.01	0.22 ± 0.02	0.79 ± 0.01
WCM w/ freeze encoder	Carrot	0.88 ± 0.02	0.47 ± 0.04	0.52 ± 0.04	0.65 ± 0.04	0.33 ± 0.05	0.27 ± 0.09	0.79 ± 0.04
	Banana	0.86 ± 0.04	0.45 ± 0.03	0.51 ± 0.04	0.66 ± 0.02	0.34 ± 0.03	0.25 ± 0.02	0.80 ± 0.03
	Pepper	0.90 ± 0.01	0.42 ± 0.07	0.51 ± 0.04	0.67 ± 0.01	0.33 ± 0.02	0.26 ± 0.05	0.79 ± 0.03
	Cloth	0.89 ± 0.02	0.44 ± 0.04	0.55 ± 0.02	0.66 ± 0.03	0.38 ± 0.03	0.29 ± 0.02	0.82 ± 0.06
	Towel	0.92 ± 0.01	0.41 ± 0.04	0.54 ± 0.03	0.71 ± 0.02	0.35 ± 0.04	0.30 ± 0.02	0.83 ± 0.03
	Stovetop	0.91 ± 0.04	0.40 ± 0.04	0.53 ± 0.01	0.70 ± 0.02	0.37 ± 0.02	0.31 ± 0.02	0.84 ± 0.06
	Sushi	0.92 ± 0.04	0.41 ± 0.01	0.54 ± 0.04	0.71 ± 0.06	0.36 ± 0.05	0.28 ± 0.01	0.85 ± 0.02
	Average	0.90 ± 0.01	0.43 ± 0.02	0.53 ± 0.01	0.68 ± 0.01	0.35 ± 0.01	0.28 ± 0.02	0.82 ± 0.02
WCM w/ inverted dynamics	Carrot	0.33 ± 0.05	0.68 ± 0.02	0.13 ± 0.04	0.27 ± 0.04	0.07 ± 0.02	0.03 ± 0.01	0.22 ± 0.03
	Banana	0.29 ± 0.02	0.71 ± 0.01	0.14 ± 0.06	0.26 ± 0.07	0.09 ± 0.02	0.03 ± 0.01	0.22 ± 0.05
	Pepper	0.31 ± 0.03	0.68 ± 0.05	0.14 ± 0.03	0.27 ± 0.01	0.08 ± 0.04	0.02 ± 0.03	0.21 ± 0.02
	Cloth	0.38 ± 0.03	0.67 ± 0.02	0.17 ± 0.03	0.31 ± 0.04	0.10 ± 0.05	0.06 ± 0.01	0.25 ± 0.06
	Towel	0.40 ± 0.01	0.66 ± 0.03	0.15 ± 0.04	0.29 ± 0.03	0.12 ± 0.02	0.05 ± 0.03	0.26 ± 0.02
	Stovetop	0.37 ± 0.02	0.63 ± 0.04	0.16 ± 0.04	0.28 ± 0.02	0.13 ± 0.04	0.07 ± 0.03	0.23 ± 0.04
	Sushi	0.37 ± 0.04	0.65 ± 0.02	0.15 ± 0.02	0.30 ± 0.04	0.11 ± 0.04	0.07 ± 0.03	0.24 ± 0.03
	Average	0.35 ± 0.01	0.67 ± 0.01	0.15 ± 0.01	0.28 ± 0.02	0.10 ± 0.01	0.05 ± 0.01	0.23 ± 0.01
SPR w/ Pred.	Carrot	0.90 ± 0.02	0.21 ± 0.02	0.55 ± 0.04	0.75 ± 0.03	0.40 ± 0.03	0.27 ± 0.05	0.90 ± 0.04
	Banana	0.90 ± 0.03	0.24 ± 0.05	0.55 ± 0.03	0.76 ± 0.04	0.39 ± 0.03	0.29 ± 0.02	0.89 ± 0.03
	Pepper	0.92 ± 0.01	0.20 ± 0.05	0.56 ± 0.04	0.74 ± 0.02	0.40 ± 0.06	0.29 ± 0.01	0.90 ± 0.01
	Cloth	0.94 ± 0.01	0.23 ± 0.04	0.58 ± 0.05	0.78 ± 0.04	0.41 ± 0.04	0.32 ± 0.03	0.93 ± 0.02
	Towel	0.97 ± 0.02	0.19 ± 0.03	0.59 ± 0.05	0.79 ± 0.03	0.42 ± 0.02	0.30 ± 0.02	0.95 ± 0.02
	Stovetop	0.96 ± 0.02	0.21 ± 0.04	0.61 ± 0.03	0.77 ± 0.03	0.43 ± 0.02	0.31 ± 0.02	0.92 ± 0.01
	Sushi	0.96 ± 0.01	0.22 ± 0.03	0.60 ± 0.02	0.78 ± 0.03	0.41 ± 0.03	0.32 ± 0.05	0.93 ± 0.03
	Average	0.94 ± 0.01	0.21 ± 0.01	0.58 ± 0.01	0.77 ± 0.01	0.41 ± 0.01	0.30 ± 0.01	0.92 ± 0.01
VLM w/ Pred.	Carrot	0.91 ± 0.01	0.40 ± 0.03	0.52 ± 0.04	0.73 ± 0.03	0.37 ± 0.02	0.27 ± 0.04	0.81 ± 0.03
	Banana	0.89 ± 0.02	0.42 ± 0.04	0.51 ± 0.05	0.74 ± 0.04	0.38 ± 0.04	0.25 ± 0.02	0.82 ± 0.03
	Pepper	0.90 ± 0.01	0.39 ± 0.03	0.50 ± 0.03	0.75 ± 0.03	0.37 ± 0.03	0.27 ± 0.05	0.82 ± 0.04
	Cloth	0.92 ± 0.02	0.39 ± 0.04	0.53 ± 0.06	0.77 ± 0.03	0.42 ± 0.06	0.30 ± 0.02	0.84 ± 0.02
	Towel	0.93 ± 0.02	0.35 ± 0.01	0.55 ± 0.02	0.76 ± 0.02	0.40 ± 0.03	0.28 ± 0.04	0.85 ± 0.06
	Stovetop	0.94 ± 0.03	0.36 ± 0.04	0.54 ± 0.02	0.78 ± 0.02	0.39 ± 0.04	0.31 ± 0.01	0.85 ± 0.05
	Sushi	0.93 ± 0.03	0.36 ± 0.02	0.56 ± 0.04	0.78 ± 0.04	0.40 ± 0.02	0.29 ± 0.03	0.87 ± 0.01
	Average	0.92 ± 0.01	0.38 ± 0.01	0.53 ± 0.02	0.76 ± 0.01	0.39 ± 0.01	0.28 ± 0.01	0.84 ± 0.01

For PPO-based experiments, we use generalized advantage estimation with discount factor $\gamma = 0.99$ and $\lambda = 0.95$. The released ManiSkill/OpenVLA-OFT configuration uses PPO clipping bounds of 0.20 and 0.28 and a value clipping coefficient of 0.20. Other environment- and policy-specific settings follow the corresponding configuration files released in the RLinf-WCM repository.

For experiments reported over multiple random seeds, all model and optimization settings are kept fixed and only the random seed is changed.

G.1 Computational Overhead of Critic Architectures

We further compare the computational overhead of different critic architectures under the same hardware and software environment. All history-aware critics use the same history length $K = 3$ and are evaluated with the same batch size and input representation. We report the number of trainable parameters, value-forward latency, training-step latency, and peak GPU memory. The training-step latency includes the forward pass, backward pass, and optimizer update. For recurrent critics, this measurement includes backpropagation through time over the observation history; for SPR, it includes both the online and EMA target branches. For WCM, the training measurement includes the value, next-latent prediction, and SIGReg objectives. At evaluation time, only the value pathway is required for WCM.

Table 8: Computational overhead of different critic architectures. All methods are measured on NVIDIA A100 GPU with history length $K = 3$ and batch size $B = 128$. Latency is mean $\pm$ standard deviation over 3 runs across different seeds.

Critic	Params (M)	Forward (ms)	Update (ms)	Peak Mem. (GB)
MLP	7.98	0.91 ± 0.02	9.31 ± 0.04	0.27
Recurrent	18.92	1.78 ± 0.06	13.61 ± 0.03	0.35
SPR	31.78	3.33 ± 0.01	22.89 ± 0.01	0.62
WCM (Ours)	25.09	2.68 ± 0.10	15.75 ± 0.17	0.49

These measurements characterize the additional computation introduced by temporal and predictive critic modeling. While WCM incurs additional cost relative to a single-frame MLP critic, it avoids recurrent unrolling and does not require a separate EMA target encoder. The prediction branch is used only during critic training and is not required for value estimation at evaluation time.

We compare the computational cost of different critic architectures under the same hardware environment, as shown in Table. 8. WCM introduces additional computation compared with a standard MLP critic due to its temporal prediction objective. However, compared with SPR, WCM requires fewer parameters and achieves lower forward latency, update latency, and peak memory usage. This indicates that WCM provides temporal predictive supervision without the additional overhead associated with separate target networks.

H LLM Usage

Large Language Models (LLMs) were used solely to assist with language polishing and wording suggestions during manuscript preparation. Specifically, LLMs were used to improve sentence phrasing, grammar, readability, and word choice. They were not involved in the research ideation, methodology, experimental design, data analysis, or formulation of scientific conclusions. All scientific content, experimental results, and final manuscript text were reviewed and remain the responsibility of the authors.