Learning Visual Feature-Based World Models via Residual Latent Action

Xinyu Zhang[1] **Zhengtong Xu**[2] **Yutian Tao**[3]
Yeping Wang[3] **Yu She**[2] **Abdeslam Boularias**[1]

[1]Rutgers University [2]Purdue University [3]University of Wisconsin-Madison

Abstract

World models predict future transitions from observations and actions. Existing works predominantly focus on image generation only. Visual feature-based world models, on the other hand, predict future visual features instead of raw video pixels, offering a promising alternative that is more efficient and less prone to hallucination. However, current feature-based approaches rely on direct regression, which leads to blurry or collapsed predictions in complex interactions, while generative modeling in high-dimensional feature spaces still remains challenging. In this work, we discover that a new type of latent action representation, which we refer to as *Residual Latent Action* (RLA), can be easily learned from DINO residuals. We also show that RLA is predictive, generalizable, and encodes temporal progression. Building on RLA, we propose *RLA World Model* (RLA-WM), which predicts RLA values via flow matching. RLA-WM outperforms both state-of-the-art feature-based and video-diffusion world models on simulation and real-world datasets, while being orders of magnitude faster than video diffusion. Furthermore, we develop two robot learning techniques that use RLA-WM to improve policy learning. The first one is a minimalist world action model with RLA that learns from actionless videos, and improves VLA on LIBERO and real robot. The second one is a visual RL framework trained entirely inside a world model learned from offline videos only, using a video-aligned reward and no online interactions.

1 Introduction

World models have recently received increasing research attention due to their great potential for policy learning and reasoning through future state prediction [1]. Currently, the predominant paradigm in world modeling relies on video generation, by predicting future trajectories in pixel-aligned VAE latent spaces [2–7]. While visually compelling, this approach is prone to hallucination [8] and suffers from a heavy computational overhead [9]. As a result, downstream applications of world models remain largely constrained to open-loop robot data generation [10, 11], policy pretraining [12, 13], and planning for specific tasks [14–16].

Visual feature-based world models predict features of future frames, such as DINO tokens, rather than just videos [17]. This direction is partly motivated by studies in cognitive science showing that humans do not reason in raw pixels but in latent spaces shaped by task goals and physical understanding [18, 19]. DINO-WM [16, 20, 21] shows that direct regression of future DINO tokens leads to efficient and accurate world models for 2D manipulation tasks. However, despite these advantages, feature-based world models remain far less adopted, as predictions often become blurry or even collapse in complex 3D interactions [10]. A seemingly straightforward solution is to use generative models in feature space. However, feature-space generation is even more difficult than in

Project page: `https://mlzxy.github.io/rla-wm`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

pixel space due to the higher dimensionality [22, 23]. More importantly, heavy generative pipelines undermine the very advantages that feature-based models should provide, as detailed in Sec. 3.

Motivated by these challenges, we seek to answer two key questions: (1) how to develop an efficient yet accurate world model in a visual feature space that scales to complex 3D manipulation? and (2) how to leverage such world models to improve downstream policies?

While visual features are high-dimensional, we believe the manifold of valid physical transitions is inherently lower-dimensional. Therefore, learning a compact representation of these low-dimensional dynamics would enable a more principled approach to visual feature-based world models. In this work, we introduce *Residual Latent Actions* (RLA). RLA is deceptively simple: it encodes the residual between DINO tokens of two frames (s_t, s_{t+h}) into a compact latent vector, and is trained with a single regression loss to reconstruct s_{t+h} from s_t, as shown in Fig. 1a. Despite its simplicity, we find that RLA exhibits three surprising empirical properties that make it well-suited for dynamics learning. (1) RLA is sufficiently predictive. As illustrated in Fig. A1, the decoder f_{dec} can accurately reconstruct s_{t+h} from RLA z and s_t in a single forward pass. In contrast, prior methods mainly use latent actions as weak conditioning labels for iterative generation [24, 25]. (2) RLA generalizes to novel scenes and motion patterns, even when trained on limited data, as shown in Fig. A5. (3) RLA latent space exhibits a temporal topology; although training is performed only on frame pairs, decoding linear interpolations between a Gaussian noise and RLA yields results that approximate intermediate frames, as illustrated in Fig. A2.

Based on RLA, we propose the *RLA World Model* (RLA-WM), shown in Fig. 2. Instead of directly regressing DINO tokens s_{t+h}, RLA-WM first predicts RLA z via flow matching with s_t and actions $a_{t:t+h}$ as input conditions, then predicts s_{t+h} from s_t and z. RLA-WM significantly outperforms state-of-the-art feature-based and video diffusion world models on both simulation and real-world datasets, while remaining more efficient as the flow matching runs in the compact RLA space.

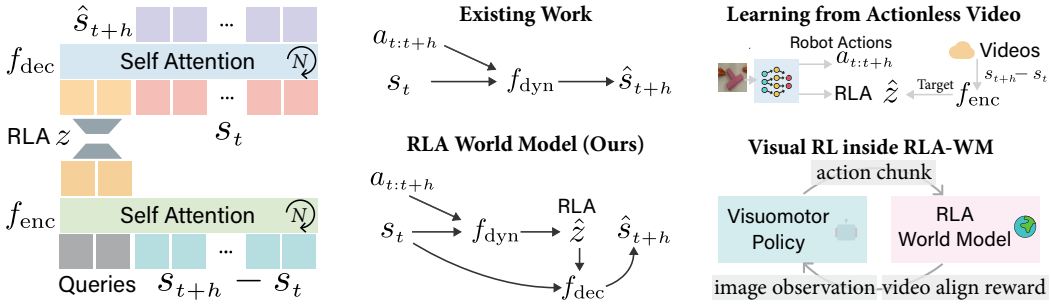
Furthermore, we introduce two robot learning techniques built on RLA and RLA-WM. First, we show that a behavior cloning policy can be extended into a minimalist world action model (WAM) using a single linear layer that predicts RLA from the current observation. Unlike prior WAMs that couple action prediction with heavy video generation backbones [7], our approach imposes no such coupling, adds no inference cost, and consistently improves policy success rates for imitation learning from actionless videos, with further gains for a VLA on LIBERO [26] and a real robot cup stacking task. Second, we perform visual reinforcement learning (RL) entirely inside a world model learned from a small offline video dataset without online interactions or even auxiliary BC loss during RL, using an alignment reward with offline videos. Our World Model-based RL (WMRL) improves the best BC policies on all five ManiSkill tasks, with statistically significant gains on the XArm and UR10e robots, and further benefits from failed trajectories that BC cannot exploit.

Our contributions are threefold. (1) We propose the Residual Latent Action (RLA), a simple latent action representation learned from DINO residuals. (2) We present RLA-WM, which predicts RLA via flow matching and sets a new state-of-the-art among visual feature-based world models. (3) We demonstrate the value of RLA and RLA-WM in two novel applications: (a) a minimalist world action model that learns from actionless videos, which also improves a VLA on LIBERO and a real-robot policy; (b) a visual reinforcement learning framework that optimizes policy via rollouts in the RLA-WM, which further improves policies from failed trajectories.

2 Related Work

World Models for Robotics. Learning world models from offline datasets has emerged as a promising paradigm for future state prediction in robotics [27, 2]. Existing approaches largely focus on predicting future videos [3–7] and 3D geometry, such as point clouds [28–31]. Despite their success, video prediction induces a heavy computational overhead due to diffusion models. While 3D world models benefit from spatial priors, their structural assumptions often limit them to specific tasks. Another line of research explores learning world models via online rollouts within simulators [32–37], but the reliance on simulators and handcrafted reward functions limits their application.

World Models in Visual Feature Space. An alternative to pixel-space prediction is embedding future states in a learned visual feature space. For instance, V-JEPA predicts future features for self-supervised learning [17, 38]. The DINO-WM family of world models [16, 20, 21] predicts DINO



(a) RLA Autoencoder Learning (b) Dynamics Learning (c) Applications

Figure 1: **Overview of our framework.** We introduce the Residual Latent Action (RLA), which compresses the DINO token residual $s_{t+h} - s_t$ into a compact latent z. We discover that RLA is predictive, generalizable, and encodes temporal progression. Next, we propose the RLA World Model (RLA-WM), which learns from offline videos and predicts RLA z instead of s_{t+h} directly. RLA-WM achieves accurate future prediction while being more efficient than state-of-the-art feature-based and video diffusion world models. Our approach enables two applications: learning policies from actionless videos and visual reinforcement learning via interaction entirely within RLA-WM.

tokens [39, 40] of future frames through a direct regression. DINO-WM [16] shows that predicting in a feature space mitigates the need for heavy generative models for 2D robot manipulation tasks. However, for complex 3D manipulation, we observe that simply applying regression in the feature space often yields blurred or collapsed estimations. In contrast, our approach avoids regression-to-the-mean, enabling efficient and accurate multi-modal prediction of DINO tokens in future frames.

Latent Actions. Learning compact latent actions from videos has emerged as a popular technique in robot learning [41–45, 24, 25, 46]. Existing approaches fall into two categories. The first leverages latent actions as proxy controls for imitation learning from actionless videos where proprioceptive data are absent [42–45]. The second utilizes latent actions as weak condition labels for video diffusion [24, 25]. In contrast, we learn Residual Latent Action (RLA) from DINO residuals instead of raw pixels. RLA outperforms existing methods [24, 45] as a better action proxy (Sec. 4.2), without requiring diffusion, and can be decoded into DINO tokens of future frames in a single feedforward pass. Different from LAPA [43], which uses latent actions to pretrain policies, RLA is designed for world modeling, i.e., predicting future states, and can be retargeted to improve policies.

3 Method

Problem Formulation. Let $x_t \in \mathbb{R}^{H\times W\times 3}$ denote the raw image observation at time t. We represent the DINO patch tokens as $s_t \in \mathbb{R}^{L\times C}$, where C is the feature dimension and $L = \frac{H\times W}{P^2}$ is the sequence length for a given patch size P. We define an action chunk of horizon h at time t as $a_{t:t+h}$. Our objective is to learn a dynamics function $f_{\text{dyn}} : (s_t, a_{t:t+h}) \mapsto s_{t+h}$ using only raw offline videos, without online rollouts or access to handcrafted reward functions or labels. This function acts as a direct, multi-step world model in the feature space.

Learning Latent Actions on DINO Residuals. The physical world is inherently uncertain, which makes the dynamics function f_{dyn} highly multi-modal. That is, given an image x_t and actions $a_{t:t+h}$, there can be multiple valid values for x_{t+h}. Prior work addresses this through generative video models to predict $\hat{x}_{t+h}$. However, these methods are computationally heavy and prone to hallucination [8]. Pioneering works such as DINO-WM and JEPA instead learn world models in a feature space, such as DINO tokens, which is more efficient, does not require diffusion, and reduces hallucination because visual features encode rich semantic and geometric information [16]. These works motivate us to design a world model that, given s_t, directly predicts $\hat{s}_{t+h}$ in DINO token space rather than predicting pixel-level $\hat{x}_{t+h}$. However, despite the impressive results of DINO-WM, the key limitation is its direct regression design, which is computationally efficient but often results in blurry or collapsed predictions in complex 3D interactions.

A straightforward solution is to revert to generative models, such as diffusion or flow matching, but in the feature space, to predict $\hat{s}_{t+h}$. However, a counter-intuitive yet critical fact is that DINO

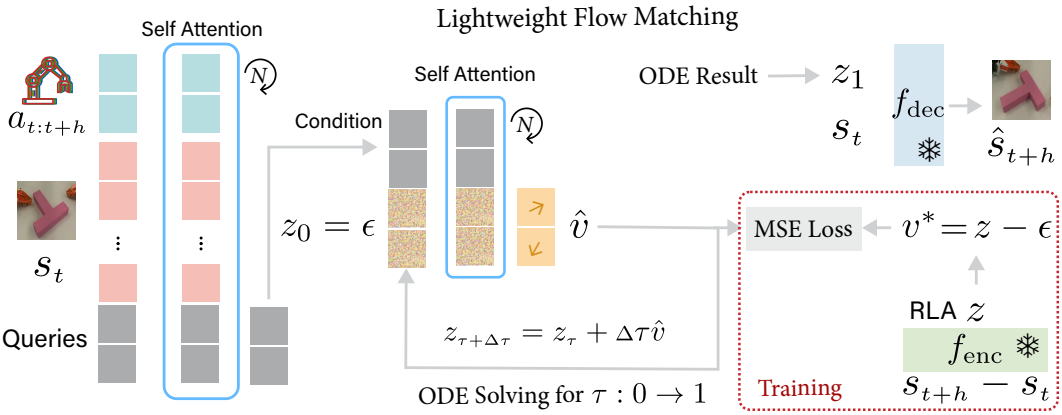


Figure 2: **RLA World Model.** RLA-WM predicts future states by generating the residual latent action z. We first embed the robot actions $a_{t:t+h}$ (padded to a maximum horizon) via an MLP. This embedding is concatenated with the DINO tokens s_t and learnable queries, then processed through self-attention layers to produce condition tokens. During flow matching, this condition is fixed and concatenated with a noisy latent z_τ, starting from Gaussian noise $z_0 = \epsilon$. The flow network predicts the velocity $\hat{v}$ to iteratively transform z_0 into the final RLA z_1. Finally, f_{dec} decodes $\hat{s}_{t+h}$ from z_1 and s_t. During training, the model is supervised by the MSE loss against the ground truth velocity $v^* = z - \epsilon$, requiring no feature or image reconstruction losses.

tokens (and ViT or ResNet features generally) have a far higher dimensionality than the pixel-aligned VAE latents used in image or video generation. For a 512×512 image, Stable Diffusion VAE [47] yields roughly $64^2 \times 4 \approx 16$k dimensions, whereas DINOv3-L tokens produce $32^2 \times 1024 \approx 1$M dimensions, nearly two orders of magnitude larger. This curse of dimensionality makes generative modeling of DINO tokens highly challenging [22]. While RAE [23] proposes diffusion techniques to generate DINO tokens from noise and class labels, it is not widely adopted because adapting it to a dynamics learning setting is not trivial, as shown in Tab. 1. More importantly, using heavy generative models defeats the purpose of feature-space learning, as they undermine both the computational efficiency and the reduced hallucination that feature-based world models offer.

To address this challenge, we shift focus from directly generating $\hat{s}_{t+h}$ to learning a representation that captures the transition from s_t to s_{t+h}. We propose learning this representation from DINO token residuals $s_{t+h} - s_t$, which also corresponds to the flow matching velocity of a Schrödinger bridge [48, 49] from s_t to s_{t+h}. Specifically, we feed these residuals along with learnable queries into an encoder f_{enc}, project the output queries to a low-dimensional space to obtain z, and pass z along with s_t into a decoder f_{dec} to reconstruct s_{t+h} (Fig. 1a). We refer to z as a *Residual Latent Action* (RLA). The RLA autoencoder uses almost only self-attention and a single regression loss on s_{t+h}. There are three key properties of RLA that set it apart from prior work, making it an ideal representation for dynamics modeling:

Predictive Sufficiency Unlike in prior work, where latent actions serve only as weak conditioning for diffusion, RLA does not require iterative generation. We find that our RLA decoder f_{dec}, when conditioned on a compact RLA z, is able to reconstruct future DINO tokens with high fidelity in a single feedforward pass. Reconstruction examples are provided in Fig. A1.

Generalizability RLA autoencoder generalizes to novel scenes. In Sec. 4.2, we demonstrate this by training RLA on task-agnostic videos and applying it to task-relevant, actionless videos for imitation learning. Examples of encoding unseen robot object interactions are provided in Fig. A5.

Temporal Topology An emergent property of RLA is the topology of its learned latent space. Although the autoencoder is only trained on frame pairs (s_t, s_{t+h}), the RLA latent space naturally encodes temporal progression. Interpolating between a Gaussian noise and RLA produces frames that correspond to temporally intermediate states, as shown in Fig. A2 and quantified in Tab. A1.

RLA World Model. Based on RLA, we revisit feature-based world modeling. Learning neural dynamics in RLA space encourages the model to capture state evolution rather than absolute states. This aligns with classical physics simulation, which models relative mesh displacements [50, 51]. Motivated by this, instead of generating high-dimensional s_{t+h}, we propose a world model to predict

Table 1: **Evaluation of Future Frame Prediction.** We evaluate RLA-WM by predicting $\hat{s}_{t+h}$ given s_t and actions $a_{t:t+h}$. We report LPIPS, SSIM, L1 distance to ground-truth DINO tokens s_{t+h} (DINO L1), and FLOPs per inference. Best results are **bold**, and second-best results are underlined.

Model	ManiSkill [52]			IWS [10]			FLOPs ↓
	LPIPS ↓	SSIM ↑	DINO L1 ↓	LPIPS ↓	SSIM ↑	DINO L1 ↓	
DINO-WM [16]	0.156	0.865	0.078	<u>0.223</u>	<u>0.825</u>	<u>0.058</u>	**2.1T**
RAE [23]	0.324	0.717	0.143	0.550	0.625	0.159	14.3T
FM-WM [53]	<u>0.127</u>	<u>0.890</u>	<u>0.063</u>	0.360	0.741	0.119	14.3T
Vid2World [6]	0.199	0.705	0.084	0.388	0.710	0.139	1.1P
RLA-WM (Ours)	**0.071**	**0.931**	**0.030**	**0.196**	**0.847**	**0.053**	<u>3.5T</u>

the compact RLA z, which is then decoded with current state s_t to reconstruct s_{t+h}. Specifically, learnable queries are concatenated with s_t and embedded actions $a_{t:t+h}$ and transformed via self-attention. These query tokens are then concatenated with a noisy RLA $z_\tau = \tau z + (1-\tau)\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, through subsequent self-attention layers to predict the velocity $\hat{v}$. During training, we supervise with ground truth velocity $v^* = z - \epsilon$, where $z = f_{\text{enc}}(s_{t+h} - s_t)$. At inference, we sample $z_0 = \epsilon$ and solve the ODE with $z_{\tau+\Delta\tau} = z_\tau + \Delta\tau\hat{v}$ from $\tau = 0$ to 1. The final z_1 is the predicted RLA, and is decoded via f_{dec} with s_t. Since the condition network is executed once and iterative generation remains within the compact RLA space, the flow matching is lightweight, as shown by the floating point operations (FLOPs) reported in Tab. 1. The compactness of the RLA space also helps the model to predict long-term dynamics more accurately, without over-attention to the excessive details in dense observation spaces. Our RLA-WM framework is illustrated in Fig. 2.

4 Experiments

Our experiments aim to answer two key questions: (1) Can RLA-WM perform accurate multi-step prediction in a visual feature space? (2) How can RLA and RLA-WM improve robot policies? To address the first question, we evaluate the RLA-WM on simulation and real-world robot manipulation videos, using image and feature prediction metrics across multi-step rollouts. For the second one, we provide two applications of RLA and RLA-WM: (a) extending behavior cloning (BC) policies to World Action Models (WAM) via RLA, and (b) performing visual RL entirely inside an RLA-WM.

4.1 Prediction Quality Evaluation

Datasets. The experiments are performed on the ManiSkill simulation suite [52] and the IWS real-world dataset [10]. In ManiSkill, we adopt three robot arms (Panda with parallel gripper, XArm with Robotiq gripper, UR10 with cylinder end-effector) across five built-in tasks: Pull Cube, Pull Cube with Tool, Roll Ball, Push T, and Poke Cube. We additionally curate a task-agnostic play environment where the robot freely interacts with primitive shapes without task-specific goals. Fig. A6 shows an overview of these environments. Unlike Dreamer [35], we do not use online interactions or rewards for training. We collect 1,000 successful and 500 failed episodes per ManiSkill task using pretrained state-based PPO agents, and 3,000 play videos per robot via scripted task and motion planning. For IWS, we select the three most challenging tasks (Push T, Rope Manipulation, and Open Box) using bimanual ALOHA robots, each providing over 600 human teleoperation demonstrations.

Training and Evaluation. RLA autoencoder is trained per dataset (ManiSkill and IWS), using a single model for multiple tasks and robots. Because each robot in ManiSkill and each scene in IWS has a different action space, as in Sec. A.2, we train the dynamics part of RLA-WM per robot on ManiSkill using task-relevant videos, and per task on IWS. For validation on ManiSkill, we use 10 success and 10 failure episodes, unseen during training, per task. For IWS, we use the official validation set. During training, we randomly sample a pair (s_t, s_{t+h}) separated by a variable horizon $h \in [1, 15]$. The network predicts $\hat{s}_{t+h}$ from s_t and actions $a_{t:t+h}$. All videos have a dimension of 512×512. During evaluation, we condition on an initial frame and autoregressively unroll predictions for 30 steps on ManiSkill (action chunk size 10) and 60 steps on IWS (chunk size 15), requiring 3 and 4 autoregressive steps, respectively. We measure the final frame's fidelity against the ground truth using LPIPS [54], SSIM [55], and the L1 distance of DINO tokens.

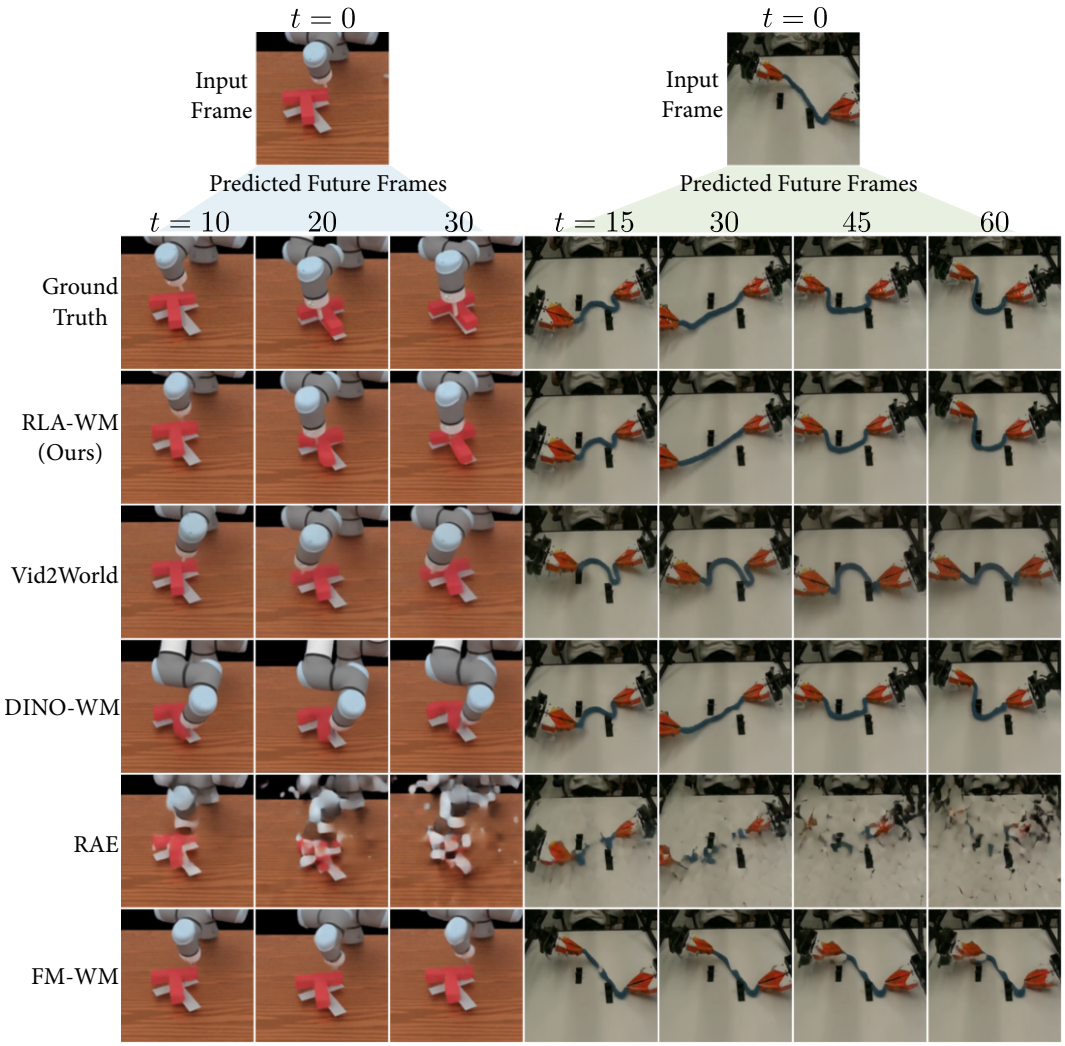


Figure 3: **Qualitative Comparison for RLA-WM.** Given an input frame at $t = 0$, our RLA-WM predicts future frames with high visual quality and physical fidelity, closely matching ground-truth states. In contrast, DINO-WM produces increasingly blurrier predictions for Push-T over longer horizons, and inconsistent rope states. Applying diffusion or flow matching directly in DINO token space yields inferior results (RAE, FM-WM). Vid2World generates visually sharp frames. However, it hallucinates and predicts physical states that diverge significantly from reality.

Baselines. We benchmark RLA-WM against a suite of state-of-the-art visual feature-based and video diffusion-based world models: (1) **DINO-WM** [16], a regression network that predicts DINO tokens. We re-implement this method using DINOv3 features to regress s_{t+h} given s_t and an action chunk $a_{t:t+h}$; (2) **Vid2World** [6], a high-fidelity video diffusion world model based on an action-conditioned DynamicCrafter [56] architecture with 1.1B trainable parameters; (3) **RAE** [23], a diffusion-based model for DINO tokens which we adapt to incorporate s_t and $a_{t:t+h}$ as conditional inputs within our transformer backbone; (4) **FM-WM**, a Flow Matching [53] baseline we implement to learn a conditional probability path that directly flows s_t to the future state s_{t+h} given $a_{t:t+h}$.

Results. As detailed in Tab. 1, RLA-WM significantly outperforms all feature-based (DINO-WM, RAE, FM-WM) and video diffusion (Vid2World) baselines across all measured metrics on both ManiSkill and IWS. We also provide qualitative comparisons on validation episodes (unseen during training) in Fig. 3 and Fig. A7 to A11. While Vid2World generates frames with sharp geometric and textural details, it hallucinates and predicts trajectories that lack physical grounding and diverge from reality, resulting in inferior metrics. Furthermore, Vid2World requires 1.1P FLOPs, a computational

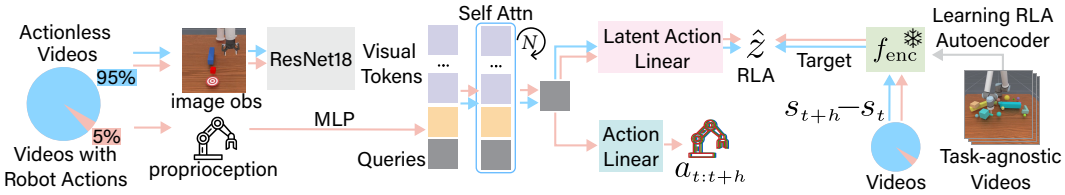


Figure 4: **Learning from Actionless Videos using RLA.** We extend a BC ResNet with a linear layer to predict the RLA $\hat{z}$. The RLA targets are extracted from (s_t, s_{t+h}) using an RLA encoder f_{enc} learned from task-agnostic videos. This turns the BC policy into a minimalist world action model that can learn from videos whose proprioceptive states and robot actions are not available, without forcing the policy to couple with DINO or video generation backbones. As shown in Tab. 2, RLA outperforms other latent action learners (replacing f_{enc}) within the same framework.

Table 2: **Latent Action Evaluation.** We report success rates (%) for imitation policies trained on actionless videos using latent actions. RLA achieves the highest average success rate and rank. Success rates are evaluated over 50 episodes (seeds 42-91) and averaged over the last five checkpoints.

Method	PushT	Roll	Pull	Pull Tool	Poke	Rank ↓	Avg SR ↑
BC-ResNet	3.6	42.0	33.6	7.6	49.2	3.8	27.2
DINO CLS [40]	7.6	39.6	40.4	4.4	44.8	4.0	27.4
UniVLA [45]	6.0	37.6	42.8	7.2	50.0	3.8	28.7
AdaWorld [24]	9.2	38.4	**48.4**	10.8	61.6	2.2	33.7
RLA (Ours)	**15.2**	**43.8**	43.6	**12.0**	**63.6**	**1.2**	**35.6**

footprint nearly three orders of magnitude larger than our 3.5T FLOPs. RLA-WM achieves high-fidelity predictions with minimal hallucination and a computational efficiency second only to the direct regression of DINO-WM. This higher performance is enabled by RLA's ability to perform flow matching within a compact latent space. Note that to compute image-space metrics, the DINO tokens $\hat{s}_{t+h}$ are decoded to RGB via a pre-trained UNet [57], which is frozen and shared by all feature-based models. Therefore, the metric gaps stem entirely from the token prediction ability.

4.2 Minimalist World Action Model with RLA

Architecture and Motivation. World Action Models (WAMs) combine robot action prediction with future video generation in a hybrid architecture [7, 58]. WAMs can be used as robot policies and show improved performance compared to policies trained with action prediction alone. However, due to the complexity of video generation, existing architectures are often tightly coupled to heavy video backbones, while predicting actions via an auxiliary module. This coupling limits their flexibility. The proposed RLA model provides a flexible alternative. We propose a minimalist WAM by extending a standard ResNet-18 [59] behavior cloning (BC) policy. We first pre-train the RLA autoencoder entirely on task-agnostic play data. The BC network then takes a 128×128 image observation and proprioceptive joint angles as input, projecting them into a shared feature. Next, the network branches into two linear heads, one predicts robot actions, and the other predicts the RLA z, which is supervised by the pre-trained RLA autoencoder. Our architecture is visualized in Fig. 4.

Learning from Actionless Video. We evaluate our minimalist WAM in a setting where only a small fraction of demonstrations contain action labels – a practical setup for scaling robot learning to large-scale, unlabeled videos. We only include robot actions and proprioceptive states for 5% of all videos (15% for PushT due to its difficulty). The remainder are actionless, video-only trajectories. During training, we construct each batch by sampling equally from videos with and without actions. For actionless videos, we mask the action loss, replace the proprioceptive input with a learnable default token, and train the shared backbone using the RLA z encoded from (s_t, s_{t+h}). During evaluation, we discard the RLA head and evaluate the policy's success rate using only the action head.

Baselines. We benchmark RLA against several latent action formulations: (1) **DINO CLS**, which uses the DINO class token of the future frame s_{t+h} as the latent action. This follows Flare [42],

Table 3: **Latent-Action VLAs on LIBERO.** We report success rates (%). We extend VLA-Adapter to also predict RLA chunks of front and wrist cameras. The policy architecture is illustrated in Fig. A3.

Method	Spatial	Object	Goal	Long	Avg
LAPA [43]	73.8	74.6	58.8	55.4	65.7
UniVLA [45]	96.5	96.8	95.6	92.0	95.2
VLA-Adapter [61]	97.0	97.4	**96.6**	**93.4**	96.1
VLA-Adapter + RLA (Ours)	**97.6**	**99.2**	**96.6**	**93.4**	**96.7**

Table 4: **Real-Robot Cup Stacking.** We train policies with 120 demonstrations on an SO-101 robot (Fig. A3), and report success over 20 rollouts.



Method	Success
BC	3/20 (15%)
BC + RLA (Ours)	**8/20 (40%)**

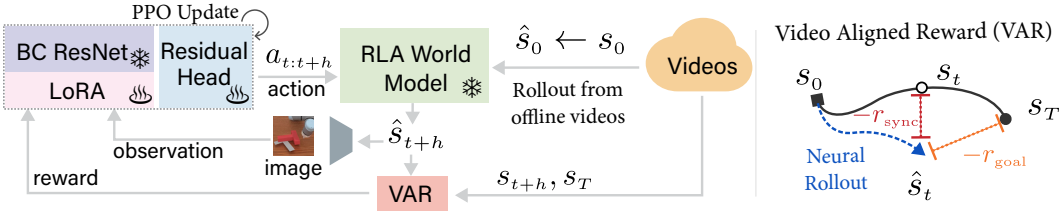


Figure 5: **Visual Reinforcement Learning within RLA World Models.** We adapt a pretrained ResNet BC policy for RL using LoRA adapters and a residual action head predicting delta actions and a Gaussian log deviation. The policy outputs action chunks $a_{t:t+h}$ to our RLA-WM, which predicts future tokens $\hat{s}_{t+h}$. A pretrained UNet decodes $\hat{s}_{t+h}$ into RGB observations for next step. RLA-WM resets its state $\hat{s}_0$ from the initial frame s_0 of a randomly sampled offline demonstration. For reward, we propose Video Aligned Reward (VAR). Because neural rollouts are time-synchronized with the reference video, VAR is simply defined as the negative DINO L1 distance between $\hat{s}_t$ and s_t (or terminal s_T). Policy is optimized via PPO using rollouts generated entirely inside RLA-WM.

which aligns visual features s_t to s_{t+h}. Note that Flare's implementation is not publicly accessible. (2) **UniVLA** [45], which learns latent actions from DINO tokens of frame pairs (s_t, s_{t+h}) using spatial-temporal attention and VQ-VAE [60]. (3) **AdaWorld** [24], which derives latent actions from frame pairs (x_t, x_{t+h}) via a VAE and operates on raw RGB images with cross-attention. For evaluation, we report success rates over 50 evaluation episodes with standard seeds from 42 to 91. Each baseline's latent extractor is pre-trained on the same dataset and substituted as the f_{enc} within our WAM framework in Fig. 4.

Results. Table 2 presents quantitative comparisons. Our RLA as a latent action significantly outperforms existing methods, achieving +8.5% over the BC baseline and +1.9% over AdaWorld (second best), yielding the highest success rate on nearly all tasks. Notably, on the most challenging Push-T task, which requires spatial reasoning and long horizons, our method shows the largest improvement (15.2% over the baseline's 3.6%), while AdaWorld (second best) achieves only 9.2%. Since all latent actions share the same auxiliary head, the gain comes from the RLA target itself instead of architecture. The unique aspect of RLA is that adding one linear layer to predict z makes the simple framework (Fig. 4) a world action model, as RLA enables accurate prediction of s_{t+h} from s_t.

Scaling to VLAs and Real Robots. We further evaluate RLA with VLA-Adapter [61] on the LIBERO benchmark [26], and compare with the latent-action VLAs LAPA [43] and UniVLA [45]. Since LIBERO includes a moving wrist camera, we extend RLA autoencoder to concatenated DINO tokens of the front and wrist cameras as in Fig. A3. VLA-Adapter predicts RLA chunks alongside robot actions. As in Tab. 3, RLA improves VLA-Adapter on Spatial and Object tasks, and achieves the highest average success rate. On a real SO-101 robot, we train BC policies for cup stacking with 120 teleoperated demonstrations from the front and wrist cameras. Adding RLA raises the success rate from 15% to 40% over 20 rollouts, as in Tab. 4. These results show that RLA remains effective with a moving wrist camera, transfers to VLA architecture and real-world robot learning.

4.3 Visual Reinforcement Learning within RLA World Model

Motivation. Extracting a robust visuomotor policy from a world model trained on offline videos remains a fundamental challenge [62]. Existing methods generally fall into two paradigms: reinforce-

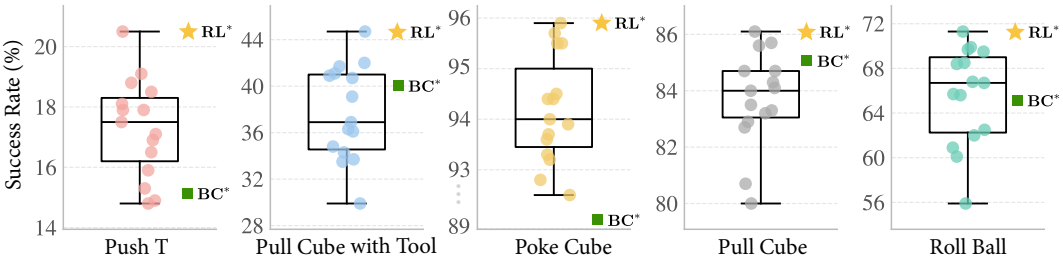


Figure 6: **World Model RL Performance Distribution.** We select the best-performing BC models (■ BC*) and apply WMRL (Fig. 5) across 15 independent seeds (1-15). Each seed's best checkpoint success rate is plotted as a dot (●), with the overall optimal checkpoint marked as ★ RL*. Success rates are evaluated on 750 episodes (seeds 1-750), according to Tab. 5.

Table 5: **World Model RL Large-Scale Evaluation.** We extensively evaluate WMRL across 750 episodes (seeds 1-750). WMRL is run with 15 seeds from the best BC checkpoint; *best* is the best checkpoint over all seeds and *avg* is the mean ± std over each seed's best checkpoint. †: WMRL (avg) outperforms BC under a one-sided paired t-test with $p < 10^{-4}$.

Method	XArm	UR10e		Panda		Avg
	Poke Cube	Roll Ball	PushT	Pull Cube	Pull Cube Tool	
BC-ResNet	89.7	65.3	15.1	85.3	40.0	59.1
RLA-WMRL (avg)	94.2† ±1.1	65.6 ±4.4	17.3† ±1.6	83.7 ±1.7	37.7 ±4.1	59.7
RLA-WMRL (best)	**95.9**	**71.3**	**20.5**	**86.1**	**44.7**	**63.7**

ment learning (RL) and planning. In RL, GWM [13] improves sample efficiency but still requires simulator interaction. UniSim [2] uses a video diffusion model as a simulator, but requires massive compute, remains unreleased, and is evaluated on a single task. Conversely, planning with world models lacks a unified framework: gradient-based methods suffer from non-convex optimization landscapes [16], and heuristic planners [63, 15] are highly task-specific. While principled methods like TD-MPC [34, 64] and the Dreamer [35] exist, they rely on pre-defined reward functions and large-scale online interactions. Thus, performing RL entirely within a learned world model is a critical open problem. This requires a world model that (1) accurately predicts the future, (2) supports efficient rollouts, and (3) operates without manual reward engineering.

Architecture. We introduce World Model-based RL (WMRL), a framework for visual RL fully within our RLA-WM. The RL policy is initialized from a pre-trained BC-ResNet policy, with LoRA adapters and a residual head added to predict Gaussian action distributions. The policy maps an RGB image at time t to an action chunk $a_{t:t+h}$, which the RLA-WM uses to predict $\hat{s}_{t+h}$. Previous works often employ SAC [65, 66] or spline [67] to ensure a strict multi-step RL formulation. We simply adopt PPO [68], treat the $(s_t, a_{t:t+h}, s_{t+h})$ tuple as a single transition, and bypass intermediate advantage estimation. We propose to rollout from offline videos by setting $\hat{s}_0$ to the first frame s_0 of a sampled demonstration. For reward, we introduce Video Aligned Reward. Since each rollout is time-aligned with a reference offline video, we compute the reward as the negative L1 distance between the DINO tokens $\hat{s}_t$ and the ground-truth s_t (or the terminal s_T). We observe a neural-to-sim gap between images decoded by UNet and the simulator's ray tracing. We reduce this gap by applying the UNet decoding as a preprocessing step. Our WMRL framework is outlined in Fig. 5.

Evaluation and Results. WMRL is an offline RL method [69] since all of the robot's actions take place inside the learned world model, without any additional interactions with the real world, which are the key cost in RL. Therefore, independent training runs can be repeatedly performed and evaluated inside the world model for free, and the best-performing policy is retained. This allows us to determine whether WMRL can surpass the performance of imitation learning, without using any additional interaction data from the real environment. Specifically, we first train the BC-ResNet policy for 40 epochs, saving checkpoints at each epoch. We use a single seed (42), as BC is generally robust to initialization and batch shuffling [70]. Then, we apply WMRL on the BC policies. To account for variance, we run 15 trials (seeds 1-15) for 2,400 steps each, saving a checkpoint every 200 updates. Instead of the standard 50 evaluation episodes, we evaluate both BC

and WMRL over 750 episodes (seeds 1-750). This directly addresses a commonly known limitation in many RL works [71, 72], where evaluations over a limited set of seeds can be unstable and prone to spurious results. As shown in Fig. 6 and Tab. 5, the best WMRL policy improves over BC on all five tasks (+4.6% on average), and the mean over 15 seeds is significantly higher than BC on the XArm and UR10e robots with $p < 10^{-4}$ under a paired t-test. Although WMRL shows lower mean performance on the Panda robot, it outperforms BC when averaged over seeds and tasks (59.7% vs. 59.1%), using no additional data and only additional computation. Each WMRL run takes about 3 hours on 4 GPUs. We discuss the Panda performance in Sec. A.4. Nevertheless, one can always devise a simple meta algorithm that selectively uses WMRL depending on the task and setup, to improve BC policies without requiring any additional data, and only returns the highest-performing policies. We view our work as a preliminary but solid step toward rigorous standards for world model-based RL.

Learning from Failures. Unlike BC, WMRL can learn from failed trajectories, since they still provide valid dynamics for the world model. We add 8,000 PushT trajectories that are executed with noisy actions and end at non-goal positions. As shown in Tab. 6, these trajectories degrade BC, since it imitates the failed behaviors, whereas we use them to train a better RLA-WM and further improves the success rate to 22.5% (best) and 18.5% (avg) through WMRL. This is a practical advantage, as failed attempts are abundant in robot learning but are typically discarded by imitation learning.

Table 6: **Learning from Failures.** WMRL improves PushT policy with 8,000 extra failed trajectories with noisy actions, which BC cannot exploit.

Method	Before	+ Failure data
BC-ResNet	15.1	13.6
RLA-WMRL (best)	20.5	**22.5**
RLA-WMRL (avg)	17.3 $\pm$1.6	**18.5** $\pm$2.3

Analysis. As the policy is optimized only inside RLA-WM, it may exploit model errors. We find that RLA-WM can hallucinate successes under noisy actions, and training with noisy data reduces this gap, as shown in Fig. A4, which is consistent with the gain from failure data. A moderate bound on the residual action head keeps the policy close to BC and consistently improves performance through WMRL, while a loose bound falls below BC, as in Tab. A2. Moreover, although VAR is time-aligned with the reference video, WMRL is not restricted to the demonstration pace. As shown in Tab. A3, we find the WMRL policy solves Poke Cube and Roll Ball with significantly shorter trajectories, even with the time-synchronized reward. Details of these studies and results are included in Appendix A.3 due to space limit.

5 Limitations and Conclusion

We introduce Residual Latent Action, a compact representation of visual state dynamics, and propose RLA-WM, a state-of-the-art visual feature-based world model. Our framework enables two robot learning techniques: a minimalist world action model, which also improves a VLA and a real-robot policy, and a visual RL framework in RLA-WM. We believe our work is a strong step forward for visual feature-based world models. However, some limitations are still worth discussing: (1) Task-irrelevant background motion can cause visual changes between s_t and s_{t+h}, which are common for humanoids or eye-in-hand cameras. Forcing RLA to encode such information could waste representation capacity and degrade the quality of RLA. Although RLA remains effective with a moving wrist camera (Sec. 4.2), large camera motion is still challenging. A solution is to learn view-independent RLA in the 3D space; (2) The visual changes between s_{t+h} and s_t can result from occlusion or partial observation, such as when the viewpoint changes, or an object disappears and later reappears. Understanding these motions requires reasoning over historical observations, which is difficult to capture from only a single frame pair (s_t, s_{t+h}). A promising solution is to extend RLA to multiple frames; (3) Our evaluation focuses on small-scale datasets and simulation, with one real-robot task. In doing so, we ensure reproducibility and isolate our method-driven gains from mere data scaling. While this demonstrates the efficacy of our method, adapting our framework to internet-scale data in an open-world setting remains an open question and promising direction. (4) WMRL relies on reference videos to compute VAR, and RLA-WM may still hallucinate successes for out-of-distribution actions. Learning rewards without aligned demonstrations and building conservative world models that are aware of their own uncertainty are promising future directions.

6 Acknowledgments and Disclosures

Funding in direct support of this work: NSF grant 2525293.

The authors acknowledge the Office of Advanced Research Computing (OARC) at Rutgers, The State University of New Jersey for providing access to the Amarel cluster and associated research computing resources that have contributed to the results reported here. URL: `https://it.rutgers.edu/research-computing`

The authors declare that they have no competing interests or financial relationships with entities that could be perceived to influence the submitted work.

References

[1] Jiahua Dong, Qi Lyu, Baichen Liu, Xudong Wang, Wenqi Liang, Duzhen Zhang, Jiahang Tu, Hongliu Li, Hanbin Zhao, Henghui Ding, et al. Learning to model the world: A survey of world models in artificial intelligence. 2026.

[2] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations (ICLR)*, 2024.

[3] Jialong Wu, Shaofeng Yin, Ningya Feng, and Mingsheng Long. Rlvr-world: Training world models with reinforcement learning. *arXiv preprint arXiv:2505.13934*, 2025.

[4] Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3d representation for consistent world modeling. *arXiv preprint arXiv:2507.07982*, 2025.

[5] Yuang Wang, Chao Wen, Haoyu Guo, Sida Peng, Minghan Qin, Hujun Bao, Xiaowei Zhou, and Ruizhen Hu. Precise action-to-video generation through visual action prompts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12713–12724, 2025.

[6] Siqiao Huang, Jialong Wu, Qixing Zhou, Shangchen Miao, and Mingsheng Long. Vid2world: Crafting video diffusion models to interactive world models. *arXiv preprint arXiv:2505.14357*, 2025.

[7] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[8] Hui Han, Siyuan Li, Jiaqi Chen, Yiwen Yuan, Yuling Wu, Yufan Deng, Chak Tou Leong, Hanwen Du, Junchen Fu, Youhua Li, et al. Video-bench: Human-aligned video generation benchmark. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18858–18868, 2025.

[9] Zangwei Zheng, Xiangyu Peng, Yuxuan Lou, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, et al. Open-sora 2.0: Training a commercial-level video generation model in $200 k. *arXiv preprint arXiv:2503.09642*, 2025.

[10] Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. *arXiv preprint arXiv:2603.08546*, 2026.

[11] GigaBrain Team, Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Haoyun Li, Jie Li, Jiagang Zhu, Lv Feng, et al. Gigabrain-0: A world model-powered vision-language-action model. *arXiv preprint arXiv:2510.19430*, 2025.

[12] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. In *Robotics: Science and Systems (RSS)*, 2023.

[13] Guanxing Lu, Baoxiong Jia, Puhao Li, Yixin Chen, Ziwei Wang, Yansong Tang, and Siyuan Huang. Gwm: Towards scalable gaussian world models for robotic manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9263–9274, 2025.

[14] Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, et al. Video language planning. In *International Conference on Learning Representations (ICLR)*, 2024.

[15] Delong Chen, Theo Moutakanni, Willy Chung, Yejin Bang, Ziwei Ji, Allen Bolourchi, and Pascale Fung. Planning with reasoning using vision language world model. *arXiv preprint arXiv:2509.02722*, 2025.

[16] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning (ICML)*, 2025.

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

[18] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[19] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the national academy of sciences*, 110(45):18327–18332, 2013.

[20] Junha Chun, Youngjoon Jeong, and Taesup Kim. Sparse imagination for efficient visual world model planning. *arXiv preprint arXiv:2506.01392*, 2025.

[21] Federico Baldassarre, Marc Szafraniec, Basile Terver, Vasil Khalidov, Francisco Massa, Yann LeCun, Patrick Labatut, Maximilian Seitzer, and Piotr Bojanowski. Back to the features: Dino as a foundation for video world models. *arXiv preprint arXiv:2507.19468*, 2025.

[22] Zhenxin Zheng and Zhenjie Zheng. Rethinking diffusion model in high dimension. *arXiv preprint arXiv:2503.08643*, 2025.

[23] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

[24] Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. In *International Conference on Machine Learning (ICML)*, 2025.

[25] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

[26] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[27] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. In *International Conference on Machine Learning (ICML)*, 2024.

[28] Tengbo Yu, Guanxing Lu, Zaijia Yang, Haoyuan Deng, Season Si Chen, Jiwen Lu, Wenbo Ding, Guoqiang Hu, Yansong Tang, and Ziwei Wang. Manigaussian++: General robotic bimanual manipulation with hierarchical gaussian world model. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12232–12239. IEEE, 2025.

[29] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[30] Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. Particleformer: A 3d point cloud world model for multi-object, multi-material robotic manipulation. *arXiv preprint arXiv:2506.23126*, 2025.

[31] Ying Chai, Litao Deng, Ruizhi Shao, Jiajun Zhang, Kangchen Lv, Liangjun Xing, Xiang Li, Hongwen Zhang, and Yebin Liu. Gaf: Gaussian action field as a 4d representation for dynamic world modeling in robotic manipulation. *arXiv preprint arXiv:2506.14135*, 2025.

[32] Chenhao Li, Andreas Krause, and Marco Hutter. Robotic world model: A neural network simulator for robust policy optimization in robotics. *arXiv preprint arXiv:2501.10100*, 2025.

[33] SV Jyothir, Siddhartha Jalagam, Yann LeCun, and Vlad Sobal. Gradient-based planning with world models. *arXiv preprint arXiv:2312.17227*, pages 703–708, 2023.

[34] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

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

[36] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

[37] Ignat Georgiev, Varun Giridhar, Nicklas Hansen, and Animesh Garg. Pwm: Policy learning with multi-task world models. In *International Conference on Learning Representations (ICLR)*, 2025.

[38] Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

[39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[40] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[41] Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? *arXiv preprint arXiv:2506.15691*, 2025.

[42] Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

[43] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations (ICLR)*, 2025.

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

[45] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

[46] Tommie Kerssies, Gabriele Berton, Ju He, Qihang Yu, Wufei Ma, Daan de Geus, Gijs Dubbelman, and Liang-Chieh Chen. A frame is worth one token: Efficient generative world modeling with delta tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27978–27988, 2026.

[47] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[48] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. *Advances in neural information processing systems*, 36:62183–62223, 2023.

[49] Valentin De Bortoli, Iryna Korshunova, Andriy Mnih, and Arnaud Doucet. Schrodinger bridge flow for unpaired data translation. *Advances in Neural Information Processing Systems*, 37:103384–103441, 2024.

[50] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, GA Pershing, Henrique Teles Maia, Maurizio M Chiaramonte, Kevin Carlberg, and Eitan Grinspun. Crom: Continuous reduced-order modeling of pdes using implicit neural representations. *arXiv preprint arXiv:2206.02607*, 2022.

[51] Yutian Tao, Maurizio Chiaramonte, and Pablo Fernandez. Interpolated adaptive linear reduced order modeling for deformation dynamics. In *Computer Graphics Forum*, page e70371. Wiley Online Library, 2026.

[52] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Cathera Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[53] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*.

[54] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[55] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[56] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Dynamicrafter: Animating open-domain images with video diffusion priors. In *European Conference on Computer Vision*, pages 399–417. Springer, 2024.

[57] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

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

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[60] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[61] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI conference on artificial intelligence*, volume 40, pages 18638–18646, 2026.

[62] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

[63] Jacob Berg, Chuning Zhu, Yanda Bao, Ishan Durugkar, and Abhishek Gupta. Semantic world models. *arXiv preprint arXiv:2510.19818*, 2025.

[64] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

[65] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[66] Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[67] Ge Li, Dong Tian, Hongyi Zhou, Xinkai Jiang, Rudolf Lioutikov, and Gerhard Neumann. Top-erl: Transformer-based off-policy episodic reinforcement learning.

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

[69] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[70] Dylan J Foster, Adam Block, and Dipendra Misra. Is behavior cloning all you need? understanding horizon in imitation learning. *Advances in Neural Information Processing Systems*, 37:120602–120666, 2024.

[71] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[72] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[73] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[74] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21469–21480, 2025.

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

[76] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[77] Charles Spearman. The proof and measurement of association between two things. 1961.

[78] Jason Y. Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *International Conference on Learning Representations*, 2024.

[79] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

A Appendix

A.1 Code

The source code of our work is included in the supplementary folder `Code`. Please see `Code/README.md` for instructions on installation, dataset setup, downloading pre-trained models, running the demo Jupyter notebook, and training.

A.2 Implementation Details

Here we summarize the details and hyperparameters for each section of our experiments. Note that we use DINOv3-Large with channel size 1024, and AdamW optimizer [73] for all experiments.

Our RLA autoencoder training code is built on TRELLIS [74], which is released under the MIT License. Our imitation learning code is built on Diffusion Policy [75], also released under the MIT License. We use pre-trained DINOv3 models [40], which are released under the DINOv3 License. We use the ManiSkill rigid-body simulation suite [52] (Apache 2.0 License; assets under CC BY-NC 4.0) and the IWS dataset [10] (MIT License).

RLA Autoencoder. We use 12 self-attention layers for both f_{enc} and f_{dec}, with 16 heads and a channel size of 1024. Input images are 512×512. During training, we randomly sample frame pairs within a horizon of 200 for the IWS dataset and 100 for ManiSkill. We use a batch size of 128, a learning rate of 10^{-4}, and train for 100k steps. Reconstruction of s_{t+h} uses both L1 and MSE loss, each with weight 1.0. On ManiSkill, we bias frame sampling toward object movement: with probability 0.9 we sample a frame pair containing object movement (object movements are pre-recorded in the simulator), otherwise we sample a random frame. On IWS (a real-world dataset), we sample frames uniformly at random. Unless specified otherwise, all experiments use an RLA size $|z| = 2048$. Specifically, the encoder output uses 32 query tokens, each projected to dimension 64. To render DINO tokens back to images, we train a separate decoder-only UNet with 4 upsampling deconvolution blocks; each block doubles the spatial resolution and halves the channel dimension.

RLA World Model. The condition network uses 8 self-attention layers with 16 heads, channel size 1024, and 32 query tokens. The flow matching network also uses 8 self-attention layers and channel size 1024, but operates on a smaller token count: only 64 tokens total (32 from the condition network and 32 for the noisy RLA z_τ), and its output dimension is 64, consistent with our RLA representation where $|z| = 2048 = 32 \times 64$. We use a maximum action horizon of 15. The RLA-WM is trained for 100k steps with learning rate 10^{-4} and batch size 64. During inference, we use 30 Euler ODE steps for flow matching. Different robots have different action sizes due to their kinematics: for ManiSkill, Panda (8), UR10 (5), XArm-Robotiq (12); for the IWS ALOHA robot, we use the dataset's provided actions — the rope task has action size 14 (7 joints for each ALOHA arm), the box task has action size 8 (3D Cartesian coordinates of each gripper, plus an additional dimension to control gripper openness), the Push-T task has has action size 4 (2D table coordinates for each of two arms). Before feeding actions into the condition network, we embed them via a robot-specific MLP: actions are first padded to the maximum horizon (e.g., for Panda with horizon 15, we pad to shape 8×15), then passed through the MLP. RLA-WM and its autoencoder each take 3 days on 4× A6000 GPUs (48GB) with 256GB RAM. The dynamics component of RLA-WM is trained per robot on ManiSkill, per scene on IWS, while the RLA autoencoder is trained per dataset.

Learning from Actionless Videos with RLA. Input images are 128×128, and we use an action chunk size of 12 during training and pick the first 8 to execute each step during inference. The visual token self-attention uses 6 layers with channel size 256 and 8 heads. Training runs for 40 epochs with batch size 64, learning rate 3×10^{-4}, weight decay 10^{-4}, and a cosine learning rate scheduler. For the BC-ResNet baseline, which uses only 5% of videos (those with actions), the epoch size is much smaller; we therefore scale up the number of epochs to match the total iterations of latent action-based training while keeping the same evaluation and checkpoint frequency (40 times). We save a checkpoint and evaluate after each epoch. Evaluation runs 50 episodes (random seeds 42-91), each with a maximum of 100 steps. Each policy is trained per task. A single training trial takes one day on an A4500 Ada GPU (24GB).

Visual RL within RLA-WM. We inject LoRA adapters into all linear and convolutional layers of the pre-trained BC ResNet policy. Unlike the architecture used in our learning from actionless video experiments, this policy does not use self-attention over spatial tokens; instead, we apply global

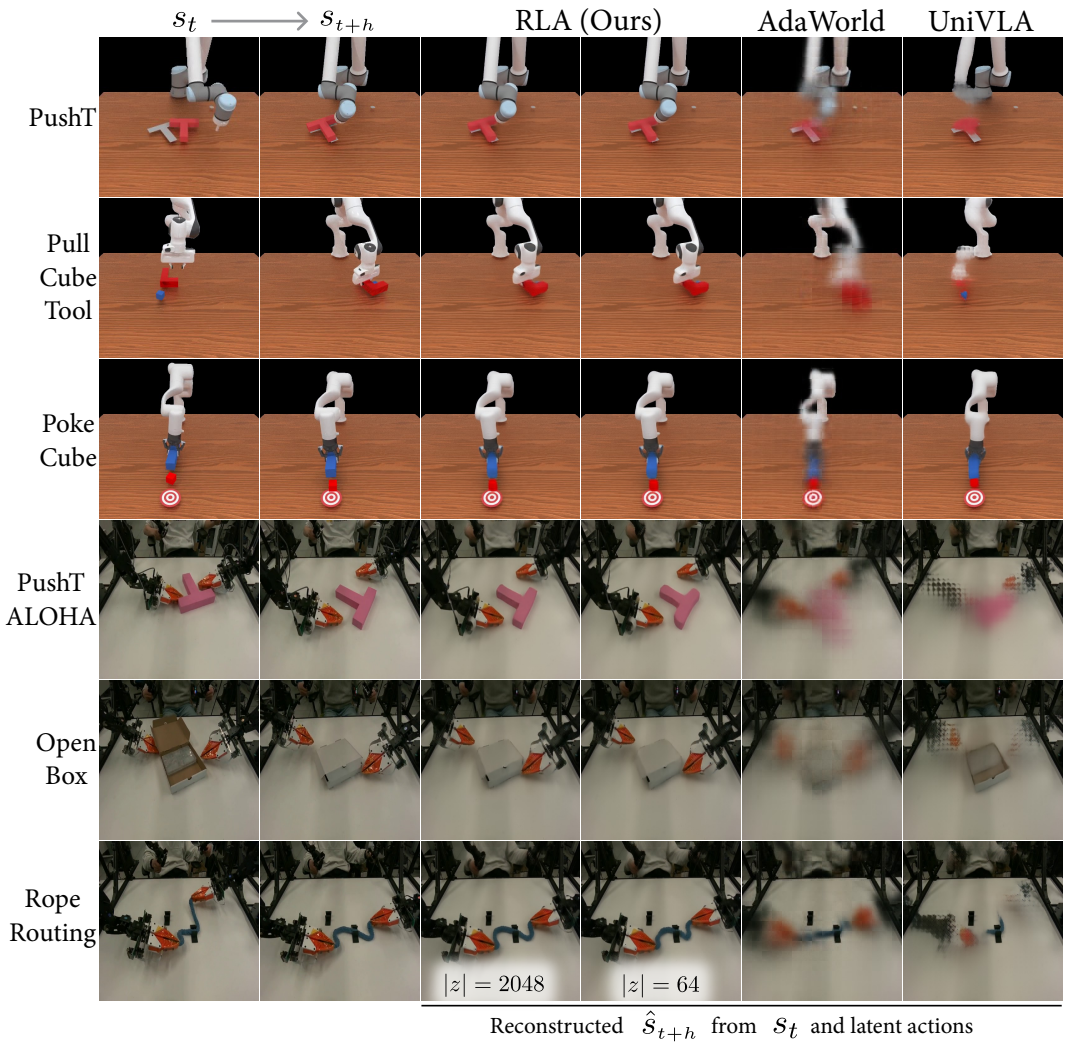


Figure A1: **Reconstruction Quality Comparison.** We compare the reconstruction $\hat{s}_{t+h}$ decoded by f_{dec} from the current frame s_t and a latent action z, where z is encoded by f_{enc} from the pair (s_t, s_{t+h}). RLA produces accurate reconstructions even with a compact latent dimension $|z| = 64$. This is particularly impressive given the 1024×1024 dimension of DINO token s_{t+h}. In contrast, AdaWorld [24] and UniVLA [45] yield severely blurred and inaccurate predictions with latent dimensions 2048 and 256, respectively. This demonstrates that RLA captures sufficient predictive information for dynamics and enables accurate future token decoding in a single feedforward pass.

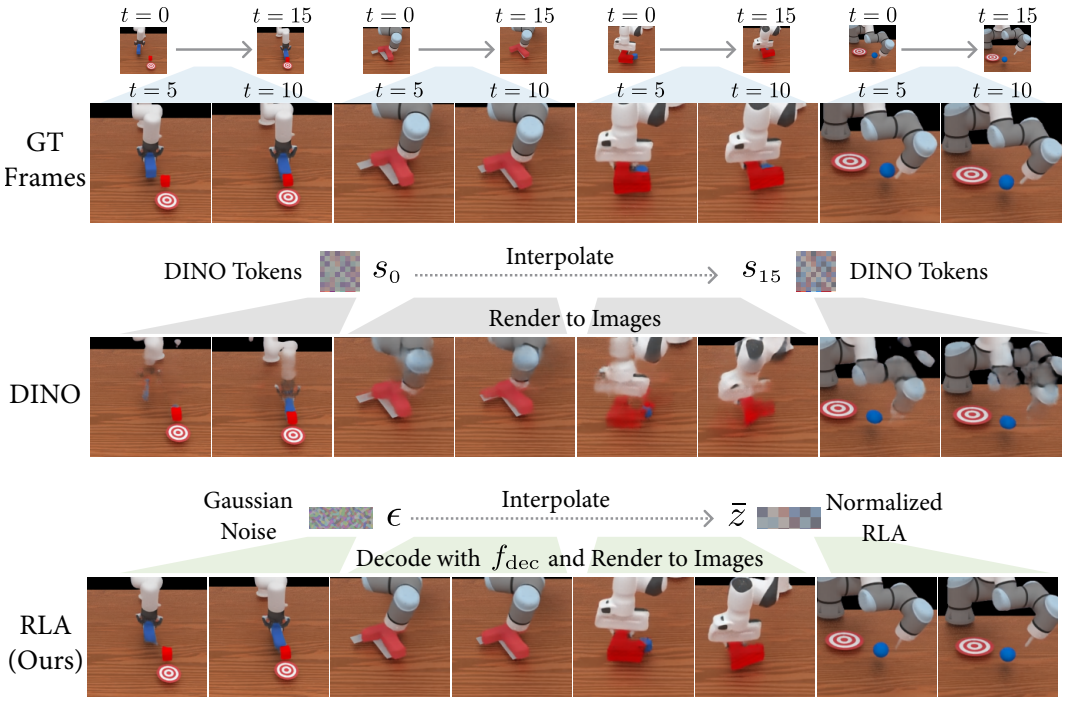


Figure A2: **RLA Temporal Topology.** We encode RLA z from (s_t, s_{t+h}) and normalize it to $\bar{z} = (z - \mu)/\sigma$, where μ and σ are estimated from data. We then interpolate between Gaussian noise ϵ and $\bar{z}$, then denormalize and decode the result. For example, $(\epsilon + \bar{z})/2$ yields a reconstruction that approximates $s_{t+h/2}$. This indicates that the RLA latent space inherently captures temporal progression. In contrast, interpolating DINO tokens between s_t and s_{t+h} produces inferior results.

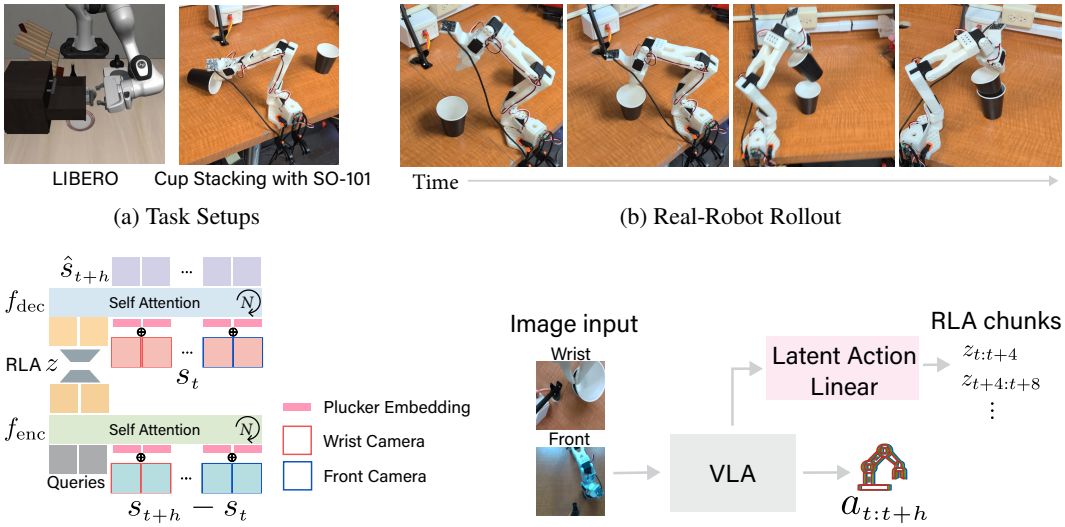
average pooling followed directly by an MLP for action prediction. The residual action head is a 3-layer MLP with hidden dimensions of 512 and 256. To stabilize exploration, the residual mean and standard deviation are bounded to 0.1 radians (joint angle) using $\tanh$ and softplus activations, respectively, which we ablate in Sec. A.3. For the Video Aligned Reward, the `Poke Cube` task uses the final goal frame s_T, while all other tasks compute the reward against the time-synchronized frame s_t. To augment initial state diversity, we initialize episodes from a random intermediate video frame with a 0.5 probability, controlled deterministically via the random seed. We apply a reward scale of 5 and rely exclusively on PPO gradients, without any terminal rewards and auxiliary behavior cloning losses.

RL optimization is conducted over 15 random seeds using a discount factor $\gamma = 0.9$, a GAE [76] parameter $\lambda = 0.95$, and a learning rate of 1×10^{-4}. We vector-parallelize 112 RLA-WM environments across multi-GPU setups. To ensure reproducibility, random seeds are uniquely mapped to the environment index rather than the GPU worker, so the same seed yields similar training results regardless of the number of GPUs. We rollout 300 steps inside the world model, each with 4 action-chunking steps of chunk size 8, and update the policy 8 times with a PPO batch size of 224. Every 200 updates we evaluate and save a checkpoint. Each training trial takes 3 hours on a 4-GPU A6000 machine or a 7-GPU A4500 Ada machine. We do not seed the Euler ODE steps for flow matching; although theoretically stochastic, we observe negligible effect on results (unlike environment resets or initial frame sampling). For final evaluation to select the best model for BC or RL, we run 750 evaluation episodes with seeds 1-750.

To bridge the *neural-to-sim* gap — the world model renders images from DINO tokens that differ slightly from simulator ray-traced images — we extract DINO tokens from each observed image, decode RGB images using our pre-trained UNet, and use the decoded image as observation. This largely removes the gap and is applied during both BC pre-training and evaluation. We acknowledge that this increases computational cost and may lower the performance ceiling, but we believe scaling the world model and policy will improve robustness to image styles in the future.

Table A1: **Temporal Correlation of Latent Interpolation.** We quantify the temporal topology of RLA (Fig. A2) on 2,000 trajectory segments whose start and end frames are 15 steps apart. We interpolate between the normalized RLA and Gaussian noise, decode each interpolant, and match it to the closest real intermediate frame. As baselines, we linearly blend the start and end frames in pixel space and in DINO token space. We report the Spearman correlation [77] between the matched and expected frame indices, and the mean frame index error. RLA interpolation follows the temporal order most closely.

	Pixel Blend	DINO Blend	RLA Interp. (Ours)
Spearman ↑	0.87	0.93	**0.97**
Frame Index Error ↓	3.13	2.51	**1.88**



(a) Task Setups (b) Real-Robot Rollout

(c) Multi-View RLA Autoencoder (d) Policy with RLA Chunk Prediction

Figure A3: **RLA for VLAs and Real Robots.** (a) We evaluate RLA on the LIBERO benchmark with VLA-Adapter (Tab. 3) and on real-world cup stacking with an SO-101 robot (Tab. 4). (b) We show a real-robot rollout, where the policy picks up a cup and stacks it onto the other cup. (c) Since both tasks use a front camera and a moving wrist camera, we extend the RLA autoencoder to multiple views. We concatenate the DINO tokens of both cameras, add a Plücker embedding [78] to the tokens of each camera, and encode their residuals into a single RLA z. (d) The policy takes the wrist and front images as input, and predicts the robot actions $a_{t:t+h}$ and, through an additional linear head, the RLA chunks $z_{t:t+4}, z_{t+4:t+8}, \ldots, z_{t+h-4:t+h}$, where $z_{t:t+4}$ is the RLA encoded from (s_t, s_{t+4}). The RLA head serves as an auxiliary training target and adds no inference cost.

A.3 Additional Experiments and Analyses

We provide additional experiments on the temporal topology of RLA, RLA for VLAs and real robots, and analyses of WMRL on model exploitation and trajectory length.

Temporal Topology of RLA. We quantify the temporal topology of RLA (Fig. A2) on 2,000 trajectory segments with horizon $h = 15$. For each segment, we encode RLA z from (s_t, s_{t+h}), interpolate between the normalized $\bar{z}$ and Gaussian noise ϵ with 11 evenly spaced coefficients over 5 noise draws, and decode each interpolant. We then match each decoded frame to the closest real frame within $s_{t:t+h}$, and report the Spearman correlation between the matched and expected frame indices, as well as the mean frame index error. As controls, we linearly blend s_t and s_{t+h} in DINO token space and in pixel space. As shown in Tab. A1, RLA interpolation yields the strongest temporal correlation (0.97) and the lowest frame index error (1.88 out of 15 frames). The same trend holds for $h = 30$, where RLA achieves a Spearman correlation of 0.97 (vs. 0.92 for DINO and 0.87 for pixel blending) and a frame index error of 4.6 (vs. 5.9 and 6.8). This temporal structure emerges from RLA despite that RLA autoencoder is only trained on frame pairs.

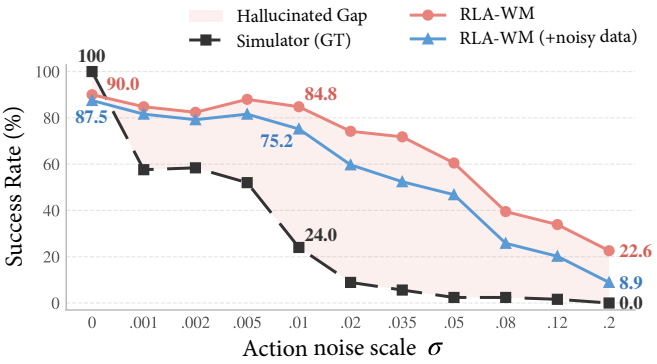


Figure A4: **Hallucinated Successes under Noisy Actions.** We replay successful PushT trajectories with drifting action noise [79] of scale σ, both in the simulator and in RLA-WM, and report the success rates (%). The shaded area is the hallucinated gap, i.e., successes predicted by RLA-WM that do not occur in the simulator. RLA-WM predicts many more successes than the simulator under moderate noise, and training it with additional noisy data narrows this gap.

Table A2: **Residual Bound and WMRL on PushT.** The residual action head outputs a mean offset $\delta_\mu = b\tanh(\cdot)$ on top of the BC policy, where $b = 0$ recovers BC and "Unbounded" removes both $\tanh$ and the bound. We report success rates (%) over 50 episodes as the mean $\pm$ std of each seed's best checkpoint over 5 seeds. A moderate bound consistently improves over BC.

Residual bound b	Success Rate
0 (BC)	18.0
0.05	**22.8** $\pm$1.8
0.1	22.0 $\pm$1.4
0.25	14.0 $\pm$3.7
Unbounded	22.7 $\pm$2.3

Table A3: **Trajectory Length of Successful Rollouts.** We report the mean number of control steps on the evaluation episodes that *both* BC and WMRL solve. Numbers in parentheses are the differences from BC, and † indicates that WMRL is shorter in at least 14 of the 15 seeds under Wilcoxon test of $p < 10^{-4}$. WMRL solves Poke Cube and Roll Ball significantly faster than BC.

Method	XArm	UR10e		Panda	
	Poke Cube	Roll Ball	PushT	Pull Cube	Pull Cube Tool
BC-ResNet	42.6	30.1	35.9	24.6	53.2
RLA-WMRL	**34.7**† (−7.9)	**23.8**† (−6.3)	36.8	25.9	**51.8** (−1.4)

Hallucinated Successes. Since WMRL optimizes the policy only inside RLA-WM, the policy may exploit model errors. To study this, we replay successful PushT trajectories with drifting Ornstein–Uhlenbeck action noise of scale σ, both in the simulator and in RLA-WM, and compare their success rates (Fig. A4). Without noise, RLA-WM reproduces most successes (90.0%). As σ increases, the success rate in the simulator quickly drops (24.0% at $\sigma = 0.01$), while RLA-WM still predicts 84.8%. That is, RLA-WM hallucinates successes for out-of-distribution actions, which the policy could exploit during RL. Training RLA-WM with additional noisy trajectories narrows this gap for all $\sigma > 0$ (e.g., from 22.6% to 8.9% at $\sigma = 0.2$), which is consistent with the improvement of WMRL from failure data (Tab. 6). Conservatism in world models remains an open problem, and we view this analysis as a first step.

Residual Bound. The residual action head outputs the mean offset $\delta_\mu = b\tanh(\cdot)$ on top of the BC policy, which bounds how far the policy can move away from BC. We vary b on PushT and run WMRL with 5 seeds (3 for unbounded). As shown in Tab. A2, a moderate bound ($b \in \{0.05, 0.1\}$) consistently improves over BC with low variance, while a loose bound ($b = 0.25$) falls below BC, likely because the policy drifts away from the BC prior and exploits model errors. Removing the bound and $\tanh$ altogether also performs well on PushT, but with a larger variance. We use $b = 0.1$ in all experiments.

Trajectory Length. VAR compares the predicted tokens with the time-synchronized reference s_t (or the goal s_T for Poke Cube), which may seem to restrict the policy to the pace of the demonstrations. To verify this, we compare the number of control steps of BC and WMRL on the evaluation episodes that both solve, paired by evaluation seed (Tab. A3). WMRL solves Poke Cube and Roll Ball with significantly shorter trajectories (-7.9 and -6.3 steps; shorter in at least 14 of 15 seeds, Wilcoxon $p < 10^{-4}$), and is comparable to BC on the other tasks. Notably, Roll Ball uses the

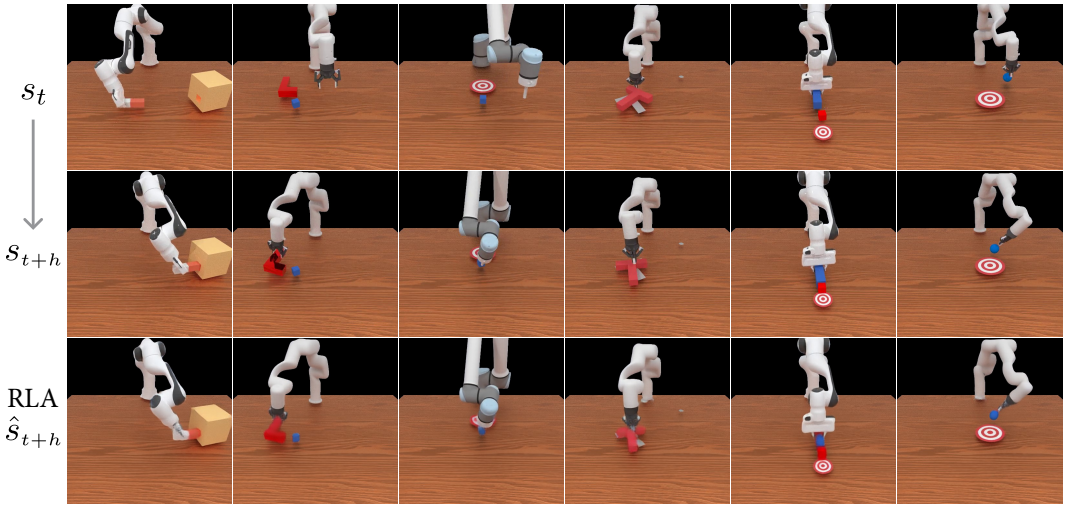


Figure A5: **RLA Generalization on Novel Tasks.** We apply the pre-trained RLA autoencoder to a previously unseen setup. For example, the original Panda robot is replaced with an XArm robot with a Robotiq gripper for the `Pull Cube with Tool` task. This interaction type was never observed during RLA autoencoder training. Yet, RLA maintains high reconstruction fidelity. Notably, this cross-embodiment generalization is achieved by training solely on the limited ManiSkill dataset, without relying on large-scale video pre-training of the RLA autoencoder.

time-synchronized reward, showing that VAR does not force the policy to track the demonstration speed.

A.4 Limitations and Future Directions

We summarize four key limitations and corresponding future directions.

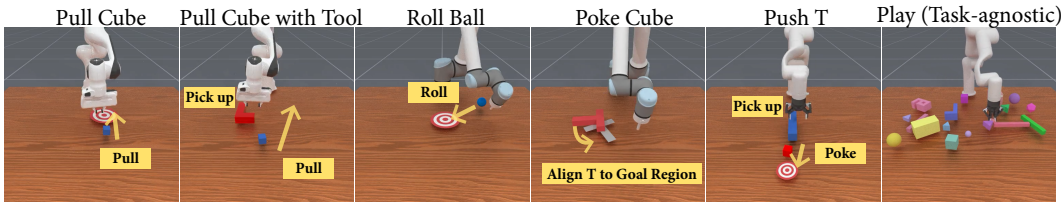
1. **Background and random motion.** Our RLA is learned from residuals between pairs of DINO tokens (s_t, s_{t+h}). However, task-irrelevant background motion or workspace randomness (e.g., in humanoid robot or eye-in-hand camera) can also cause visual changes between s_t and s_{t+h}. Learning to encode those randomness-driven motions could waste representation capacity and degrade the RLA latent space. A promising fix is to move from 2D image learning to 3D, projecting DINO tokens into 3D [74] and learning view-independent 3D RLA.
2. **Memory and partial observability.** Our RLA-WM predicts s_{t+h} from s_t and $a_{t:t+h}$, yet changes may depend on $s_{<t}$ due to occlusion (e.g., an object disappears and reappears). Because RLA z is learned from a single frame pair, it must memorize the object in the latent space rather than encoding true movement and occlusion events. Extending RLA to condition on multiple frames is a natural solution.
3. **Proprioceptive world model.** Our RLA-WM predicts only visual state evolution via RLA, but not future proprioceptive states. Proprioception input has been shown to be useful for policy learning. Extending the world model to predict both would broaden applicability.
4. **Scaling to larger datasets.** We deliberately evaluated on small-scale ManiSkill and IWS datasets to isolate method-driven gains from mere data scaling — many prior works scale first and leave it unclear whether improvements come from data volume or the method itself. Our clear, reproducible results on small data demonstrate the core properties of RLA and RLA-WM. Therefore, scaling to massive real-world datasets is a promising future step.

Panda WMRL Results. WMRL underperforms BC on the Panda robot for the `Pull Cube` and `Pull Cube with Tool` tasks. We train the dynamics component of RLA-WM per robot, yet only the Panda results show a consistent drop. We attribute this to a combination of factors: (1) the Panda's kinematic structure, (2) the camera viewpoint, and (3) insufficient action diversity in the demonstration data.

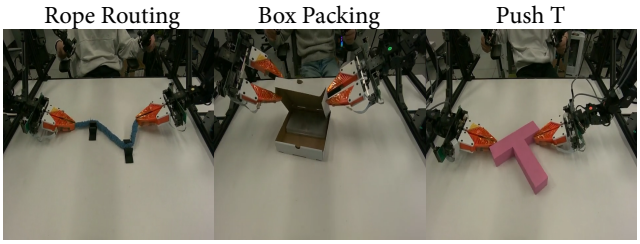
Table A4: Detailed Per-Robot Evaluation of Future Frame Prediction on ManiSkill. Our RLA-WM achieves the best results across all robots and metrics.

Model	LPIPS ↓			SSIM ↑			DINO L1 ↓		
	Panda	XArm	UR10	Panda	XArm	UR10	Panda	XArm	UR10
DINO-WM	0.187	0.120	0.163	0.841	0.896	0.857	0.098	0.056	0.079
RAE	0.310	0.313	0.349	0.719	0.730	0.701	0.139	0.166	0.124
FM-WM	0.147	0.119	0.114	0.872	0.900	0.898	0.075	0.059	0.054
Vid2World	0.219	0.186	0.192	0.685	0.720	0.710	0.094	0.081	0.077
RLA-WM (Ours)	**0.091**	**0.061**	**0.061**	**0.912**	**0.941**	**0.941**	**0.035**	**0.028**	**0.027**

The Panda arm has 8 degrees of freedom (DoF), whereas the XArm has 7 (the Robotiq gripper consists of 6 correlated joints that provide 1 effective DoF), and the UR10e has 5 (the sixth joint is ineffective due to a cylindrical end-effector). Since our policy predicts future joint angles, higher-dimensional action spaces naturally require more data. However, all robots receive the same number of demonstrations. Moreover, the two tasks of Panda both involve pulling a cube toward the robot's side of the table, producing action patterns with limited diversity. This pulling motion frequently causes occlusion from our front top-view camera, hindering the world model's ability to capture the mapping from joint angles to visual changes accurately. Additionally, the Panda's gripper fingers are small, making it difficult for the world model to capture fine visual details. We believe that adding multiple camera views, increasing task variety (and thus data diversity), and collecting more demonstrations overall would resolve these issues.



(a) ManiSkill Environment [52]



(b) IWS Dataset [10]

Figure A6: **Overview of Tasks and Datasets.** ManiSkill includes five tasks across three robots: Panda — `Pull Cube` (pull cube to target area) and `Pull Cube with Tool` (grasp L-shaped tool to hook cube within a distance to the robot); UR10 with cylinder end-effector `Roll Ball` (move ball to target region) and `Push T` (align T-shaped object with T-shaped goal area); XArm with Robotiq gripper `Poke Cube` (pick blue stick and poke cube to target). IWS (ALOHA robot) includes `Rope Routing` (route rope around marked anchors on table), `Box Packing` (open, close, or move the box), and `Push T` (continuous two-arm interaction with the T-object to create diverse movements).

Table A5: Detailed Per-Task Evaluation of Future Frame Prediction on IWS. Our RLA-WM achieves the best results on all tasks except rope routing, where it performs within a small margin of the best.

Model	LPIPS ↓			SSIM ↑			DINO L1 ↓		
	Box	PushT	Rope	Box	PushT	Rope	Box	PushT	Rope
DINO-WM	0.223	0.279	**0.167**	0.830	0.774	**0.870**	0.055	0.077	**0.043**
RAE	0.596	0.523	0.530	0.549	0.669	0.656	0.167	0.153	0.157
FM-WM	0.347	0.373	0.359	0.750	0.735	0.738	0.115	0.121	0.119
Vid2World	0.384	0.398	0.382	0.716	0.702	0.714	0.134	0.142	0.142
RLA-WM (Ours)	**0.206**	**0.204**	0.177	**0.839**	**0.845**	0.857	**0.055**	**0.055**	0.048

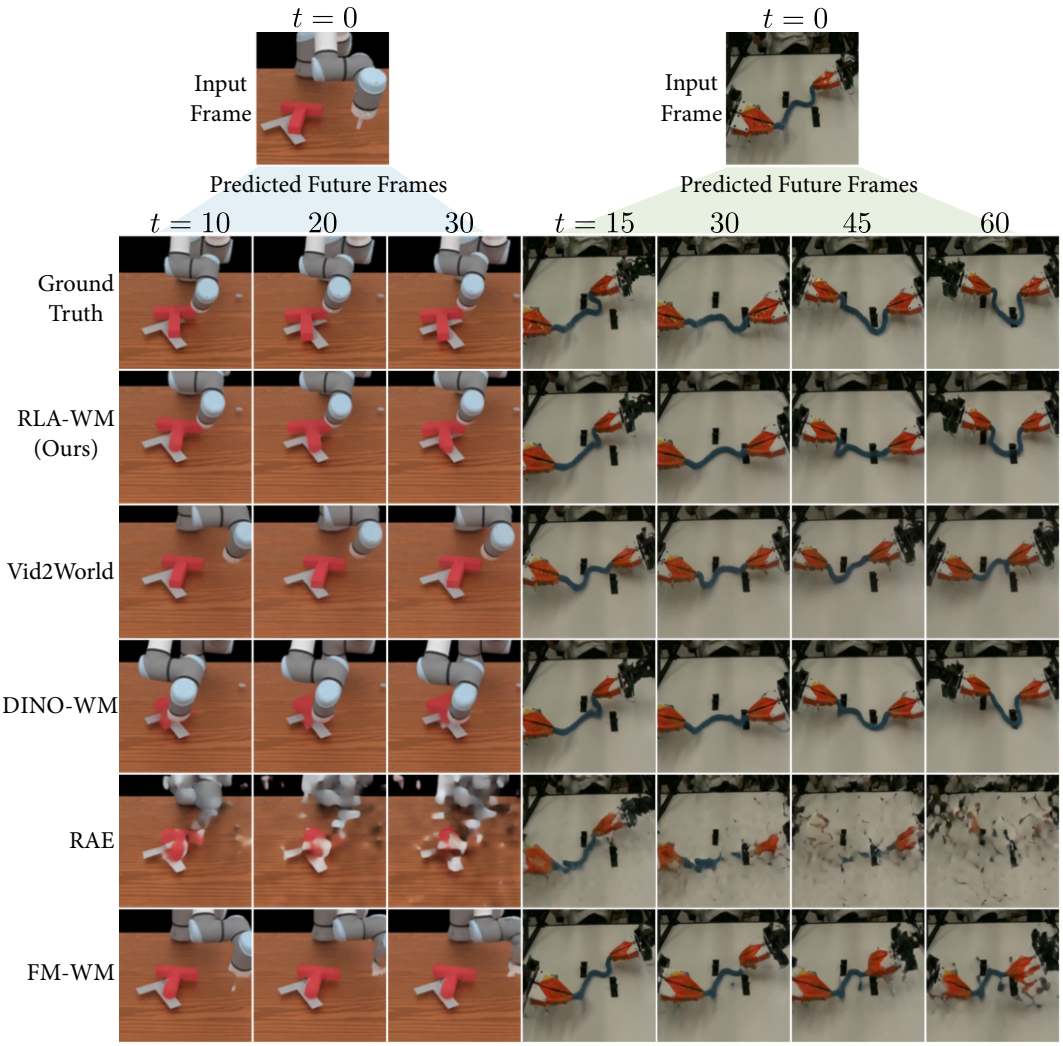


Figure A7: **Additional Qualitative Comparison for RLA-WM.**

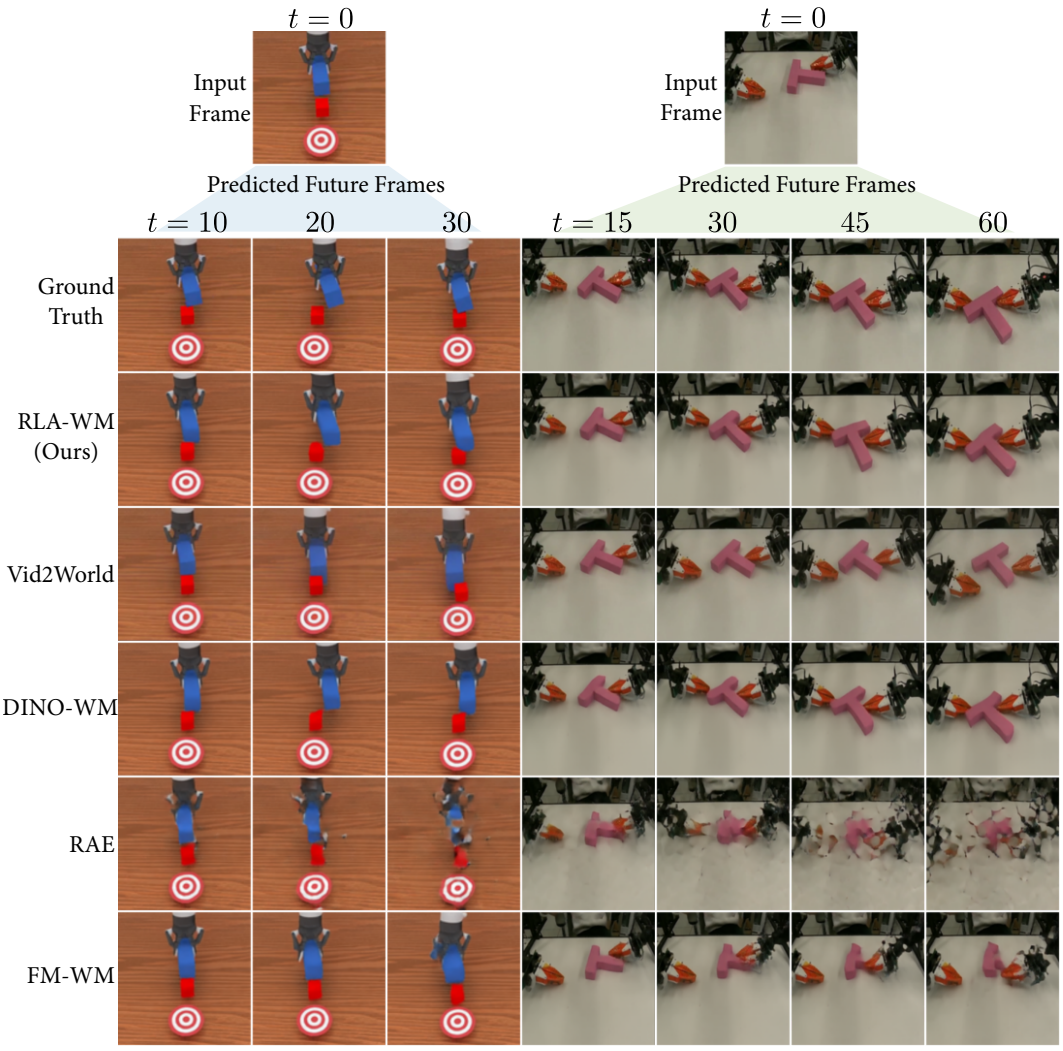


Figure A8: **Additional Qualitative Comparison for RLA-WM.**

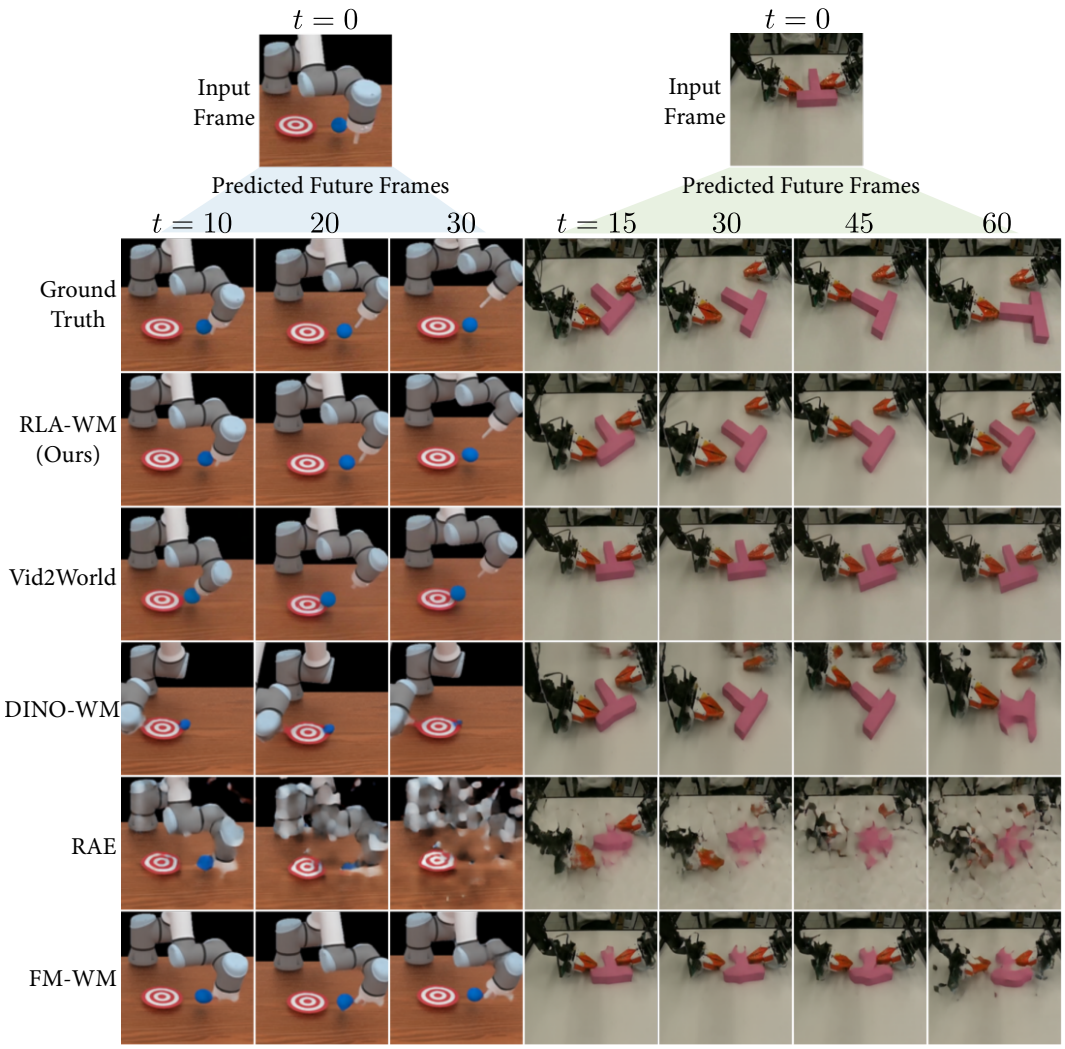


Figure A9: **Additional Qualitative Comparison for RLA-WM.**

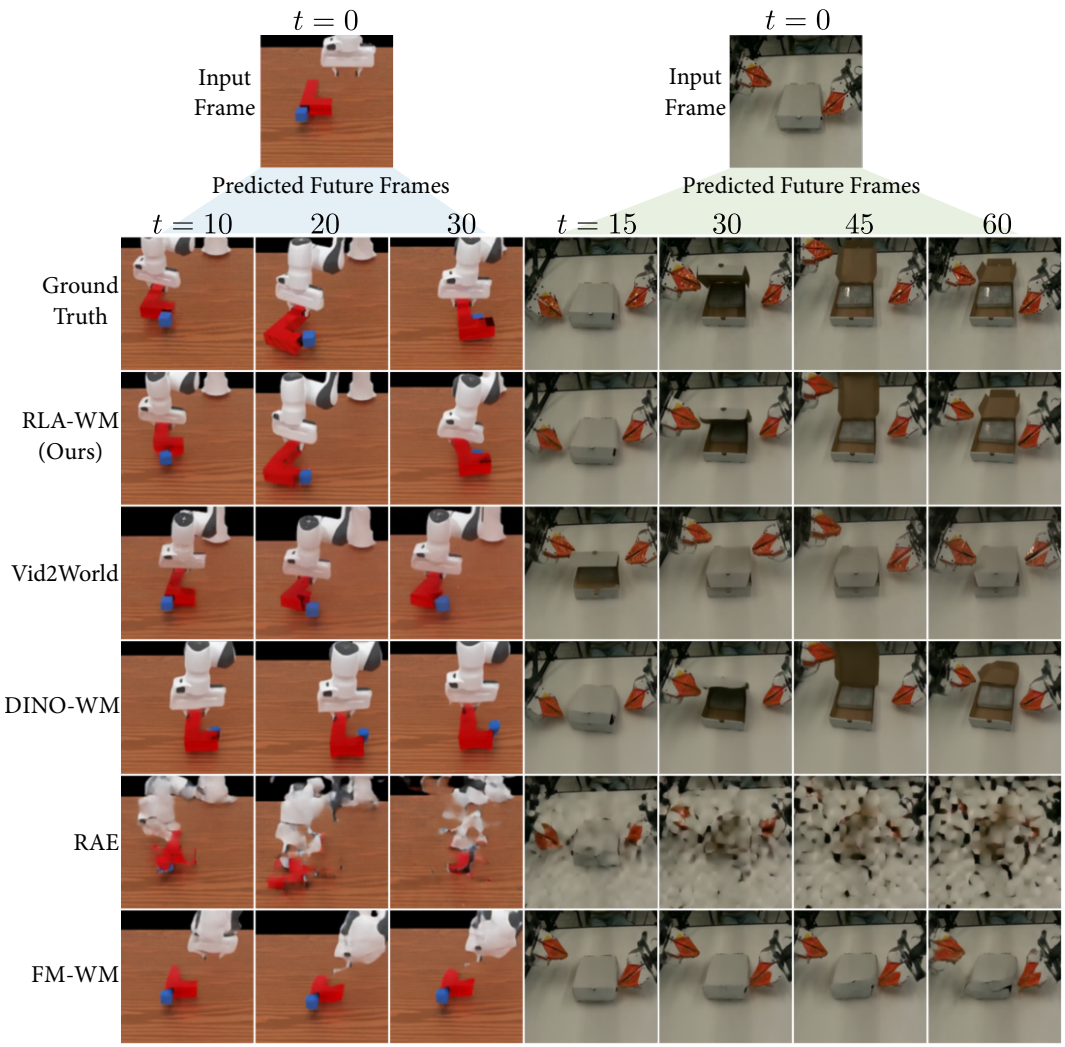


Figure A10: **Additional Qualitative Comparison for RLA-WM.**

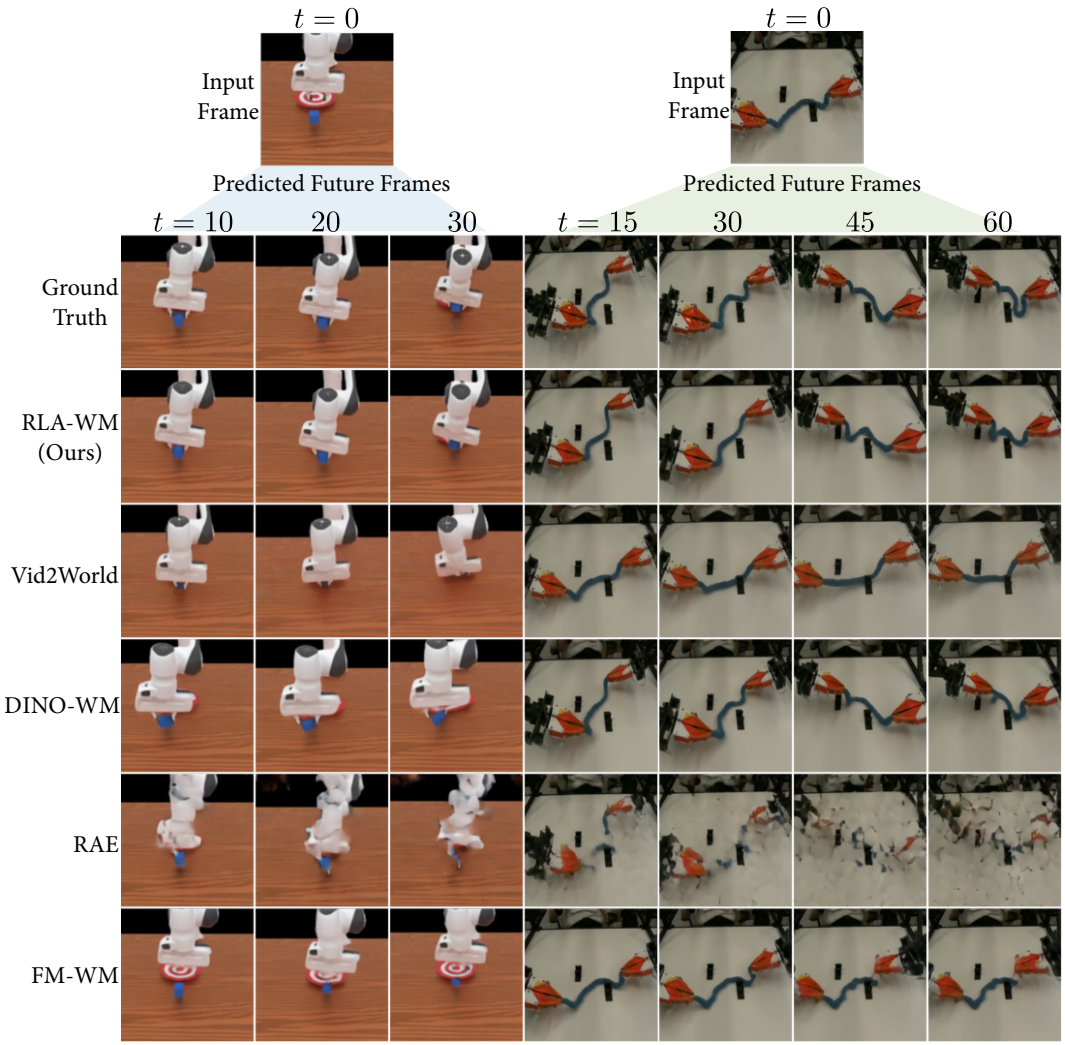


Figure A11: **Additional Qualitative Comparison for RLA-WM.**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Sec. 3 and 4 provide concrete theoretical and empirical evidence supporting the contributions claimed in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss the limitations of our proposed method in Sec. 5 and A.4.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This work focuses on empirical methodology and experimental evaluation; it does not introduce new theoretical proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Comprehensive implementation details, including network architectures and hyperparameter configurations, are provided in Sec. A.2.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Anonymized source code is included in the supplementary material for reproducibility. We will open-source the codebase upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full experimental settings, including dataset splits, training procedures, and baselines, are detailed across Sec. 4.1, 4.2, 4.3, and A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Following standard practice, we report the average or distribution across multiple checkpoints and trials, as in Tab. 2, Fig. 6, and Tab. 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed hardware specifications and estimated compute hours for all experiments are provided in Appendix A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research fully complies with the NeurIPS Code of Ethics. We utilize standard public datasets without human subjects and foresee no direct paths to societal harm.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work advances latent dynamics modeling for robotic manipulation. We do not foresee any direct negative societal impacts arising from this methodology.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: As this work focuses on feature-space dynamics modeling for robot manipulation and relies on standard offline datasets rather than scraped or generative media, it does not pose high risks requiring specific release safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and software packages utilized are appropriately cited, and their respective licenses are documented in Sec. A.2.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Documentation and usage instructions for our proposed codebase are provided in the `Code/README.md` within the supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research utilizes entirely offline robotic datasets and does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Not applicable, as this research does not involve human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as a methodological component in this research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.