DiPA-WM: Diffusion Policy Adaptation with Continual World Modeling

Anonymous authors
Paper under double-blind review

Abstract

World models enable sample-efficient policy learning by allowing agents to optimize action decisions through rollouts based on physical environment representation. In the offline-to-online reinforcement learning setting, the data distribution often shifts as the policy improves, requiring the world model to continuously adapt to on-policy data. This creates a fundamental challenge for diffusion-based policies, whose denoising process depends on stable conditioning representations: updating the world model may cause *representation drift*, while freezing it prevents tracking the evolving policy distribution. Observing the world model consists of a *context encoder* and a *dynamics adapter*, we propose **DiPA-WM** (**Di**ffusion **P**olicy **A**daptation with Continual **W**orld **M**odeling), which addresses this tension through a factorized two-stage design, outlined as follows. In Stage 1, the world model is pretrained on offline data and a 1-step Gaussian policy is learned over imagination; and the context encoder is frozen after Stage 1 training. In Stage 2, a diffusion policy is initialized via behavior cloning and then trained further over imagination while the adapter is updated on freshly collected on-policy data. The frozen context encoder provides stationary conditioning features, mitigating representation drift; the world model adapter enables continual adaptation of the world model dynamics to the evolving policy distribution. To address possible non-stationarity, we further develop a nested two-loop adaptation for the world model and diffusion policy, facilitated by periodic updates of context encoder on a larger timescale. DiPA-WM is modular and general: it consistently improves multiple diffusion policy optimization methods across continuous locomotion, robotic manipulation, and autonomous driving benchmarks, outperforming frozen-world-model and model-free baselines without modifying any core training objectives.

1 Introduction

World model learning (1; 2) offers a powerful substrate for sample-efficient policy optimization. By learning a compact latent representation of environment dynamics, agents can evaluate candidate policies through imagined rollouts rather than costly environment interactions. In the offline-to-online reinforcement learning setting, where a world model is pretrained on offline demonstrations and then jointly refined with online experience, this paradigm is particularly compelling: offline data provides broad coverage of the state space, while on-policy data enables both the model and the policy to improve together as the agent explores. The diversity and fidelity of imagined rollouts scales with the model's capacity (3), making world models an especially attractive training substrate for expressive, high-capacity policies.

Diffusion policies have emerged as a widely adopted high-capacity class for generative control tasks (4; 5; 6). By modeling the action distribution as an iterative denoising process or a learned continuous flow, they can represent complex, multimodal behaviors that unimodal policies cannot. The diverse imagined trajectories produced by a world model provide exactly the varied training signal that diffusion models require to learn well-calibrated conditional action distributions, making the combination of world models and diffusion policies a natural and promising direction for model-based reinforcement learning.

However, naively integrating diffusion policies with a continually adapting world model creates a fundamental training instability (as evidenced in the experimental studies in Appendix C). As the

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

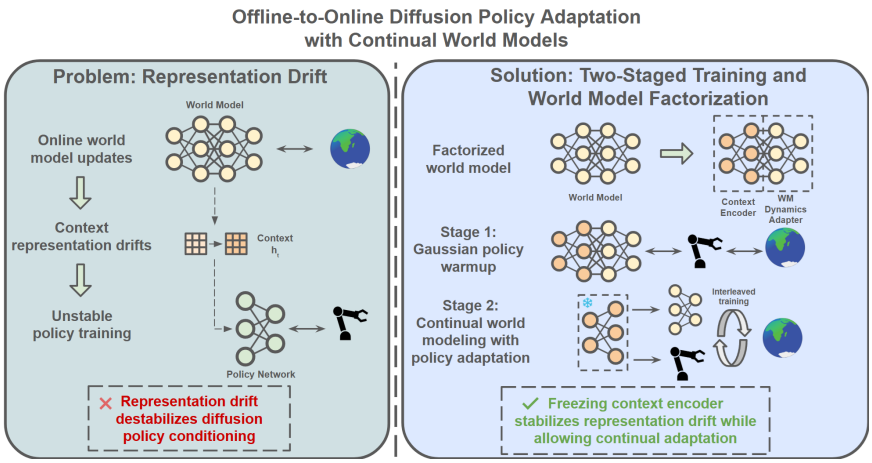


Figure 1: Overview of DiPA-WM for diffusion policy adaptation with continual world models. As the world model is updated using online, on-policy data, the latent conditioning representation h_t can drift over time, which may destabilize diffusion policy training. DiPA-WM addresses this by leveraging the factorized design of the world model (WM) which consists of a context encoder and a WM dynamics adapter: in Stage 1, the world model and a Gaussian policy are jointly trained to learn a stable latent representation from both offline and on-policy interaction; in Stage 2, the context encoder is frozen while the diffusion policy and world model adapter continue to adapt using on-policy data, thereby preserveing a stationary latent conditioning space for the diffusion policy while allowing the world model to continually improve.

policy improves, it visits new regions of the state-action space, requiring the world model to update on the resulting on-policy data. Such updates may alter the latent representation produced by the world model, including the context representations that condition the diffusion policy's denoising process. Because diffusion models learn a conditional generative mapping (7; 8), they are acutely sensitive to non-stationarity in their conditioning distribution: shifts in the latent state during training disrupt the learned conditional vector field, causing what we term *representation drift*. The two natural remedies are both unsatisfactory: jointly updating the world model and diffusion policy leads to instability, while freezing the world model prevents it from adapting to the evolving policy distribution and caps asymptotic performance.

To address this challenge, we propose Diffusion Policy Adaptation with Continual World Modeling (**DiPA-WM**), a training framework that alleviates this tension by leveraging the structure of the world model. Our core observation is that the world model serves two functions that place fundamentally different demands on learning stability: encoding histories into compact representations and predicting future dynamics (this will be elaborated in Section 3). The context encoder in Eqn. 2, which maps observation histories to latent states, primarily captures the semantic structure of the environment; it can be reliably trained from offline data and early on-policy experience, and need not change as the policy improves further. The WM dynamics adapter in Eqn. 3, which predicts future trajectories conditioned on these latent states, must track the evolving policy distribution and requires continual updating. We exploit this asymmetry: after the initial training stage, the context encoder is frozen and only the adapter is updated online. The frozen encoder guarantees that the diffusion policy's conditioning inputs remain stationary throughout training, preventing representation drift from the encoder, while the adapter keeps the world model responsive to on-policy data.

Built on this observation, DiPA-WM proceeds in two stages (Figure 1). In Stage 1, we pretrain the full world model on offline data and train a Gaussian policy in imagination, obtaining a well-calibrated world model and a capable initial policy. In Stage 2, we freeze the context encoder, initialize a diffusion policy via behavior cloning from the Gaussian policy's replay buffer, and train the two in an interleaved loop: the diffusion policy is optimized entirely in imagination, while the dynamics adapter is updated on freshly collected on-policy data. This decoupled update schedule is the core of DiPA-WM, enabling simultaneous improvement of both the policy and the world model without introducing representation drift. Furthermore, to address possible non-stationary in the environment, we expand Stage 2 to nested two-loop adaptation of world model and diffusion policy, as illustrated in Fig. 2. In particular, the policy and dynamics adapter are trained in an interleaved manner in the inner loop, whereas the context encoder is updated periodically on a fixed schedule in the outer loop.

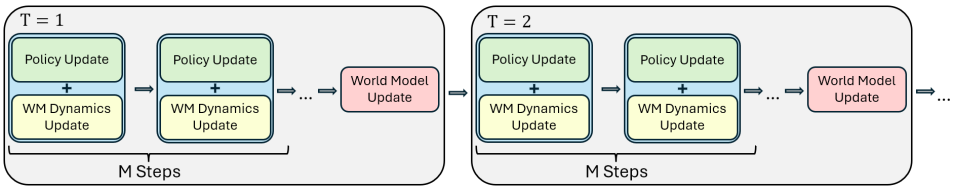


Figure 2: Nested Two-Loop Adaptation of World Model and Diffusion Policy. The policy and dynamics adapter are trained in an interleaved manner in the inner loop, whereas the context encoder is updated periodically on a fixed schedule in the outer loop.

We evaluate DiPA-WM on four benchmark suites: MuJoCo locomotion (9; 10), FrankaKitchen (11) and RoboMimic robotic manipulation (12), and CARLA autonomous navigation (13). Applied to three diffusion policy optimization methods (DPPO (14), FPO (15), and DSRL (16)), DiPA-WM consistently and substantially improves over both frozen-world-model baselines and model-free approaches, without modifying any core policy optimization objectives. Gains are especially pronounced in complex, long-horizon environments such as CARLA, where continual world model adaptation provides the greatest benefit.

Our contributions are as follows:

- We identify *representation drift*, the instability arising when a continually updated world model perturbs the conditioning inputs of a diffusion policy, as a fundamental obstacle to offline-to-online diffusion policy learning in world models.
- We propose a factorized world model design and a two-stage offline-to-online training procedure comprising a frozen context encoder and a trainable dynamics adapter, which prevents representation drift from the encoder while preserving the model's capacity to track on-policy data.
- We introduce a nested two-loop adaptation training regime to enable joint continual learning of the full world model and diffusion policy.
- We demonstrate across four benchmarks and three diffusion policy optimization methods that DiPA-WM consistently improves performance without modifying existing policy objectives, confirming the generality and modularity of our approach.

2 Related Work

World Models World models learn predictive representations of environment dynamics that enable policy optimization through imagined rollouts (17; 18; 1). Modern latent world models compress high-dimensional observations into compact latent states and learn dynamics directly in latent space, enabling scalable reinforcement learning from visual observations. Early latent world models primarily relied on recurrent architectures to summarize interaction history into a latent recurrent state (19; 20; 2). These approaches demonstrated strong performance on long-horizon control tasks and large-scale environments, including solving complex tasks such as diamond collection in Minecraft (2).

More recent work has explored increasingly expressive sequence modeling architectures for world modeling, including transformers (21; 22; 23; 24) and diffusion or flow-based generative models (25; 26; 27). These models improve long-horizon prediction quality and enable more expressive trajectory generation in latent space. In addition, world-model-based reinforcement learning naturally supports continual improvement of the dynamics model using on-policy data collected by the evolving policy.

Reinforcement learning for diffusion policies Diffusion-based policies have emerged as a powerful class of generative models for control, capable of representing complex and multimodal action distributions (4; 5; 6). Early work primarily focused on imitation learning and offline trajectory generation, where diffusion models were trained using behavior cloning objectives without explicit reward optimization (28). More recently, motivated by the success of reinforcement learning fine-tuning for generative models in domains such as text-to-image generation (29; 30; 31; 32), several works have explored reinforcement learning for diffusion-based control policies (33; 34).

In robotics and continuous control, existing approaches have largely focused on on-policy optimization of diffusion policies. Methods such as DPPO (14) and ReinFlow (35) formulate the denoising or flow-matching process as a sequential decision problem, treating each denoising step as a Gaussian policy optimized using PPO (36). FPO (15) instead exploits the structure of flow matching to derive a surrogate PPO objective that directly optimizes the continuous flow dynamics without requiring exact likelihood computation. Other approaches, such as DSRL (16), optimize diffusion policies indirectly by performing reinforcement learning in the latent-noise space of a pretrained policy, while off-policy approaches have also been explored as well (37).

Training diffusion policies in world models Combining diffusion policies with learned world models is a natural direction for enabling expressive decision-making in model-based reinforcement learning. Recent approaches have explored optimizing diffusion policies inside pretrained latent world models using imagined rollouts (38; 39; 40; 41). In these methods, the world model is typically trained offline and then frozen during policy optimization in order to avoid instability caused by changing latent dynamics and conditioning representations. For example, DiWA (40) performs offline adaptation of diffusion policies inside a pretrained world model using imagined trajectories.

Our work instead studies continual world model learning, where the model continues to improve using on-policy data while the diffusion policy is optimized in imagination. We focus specifically on the instability introduced by representation drift in the shared latent conditioning space. To address this, we decompose the world model into a frozen context encoder and a trainable dynamics adapter, allowing the policy to operate on stable conditioning features while the world model remains adaptive to newly collected data.

3 Problem formulation

Markov decision process. We consider sequential decision making in a Markov decision process $\mathcal{M} = (\mathcal{X}, \mathcal{A}, P_0, P, R, \gamma)$, with agent-observable states $x_t \in \mathcal{X}$ (which may include images, proprioceptive readings, or a short observation history), actions $a_t \in \mathcal{A}$, initial-state distribution P_0, transition kernel $P(x_{t+1} \mid x_t, a_t)$, reward $R(x_t, a_t) \in \mathbb{R}$, and discount $\gamma \in [0, 1)$. We assume access to an offline dataset $\mathcal{D}_{\text{off}}$ of environment trajectories. A policy π induces a trajectory $\tau = (x_0, a_0, x_1, a_1, \ldots, x_T, a_T)$ with $x_0 \sim P_0$ and $x_{t+1} \sim P(\cdot \mid x_t, a_t)$, and we seek to maximize the expected discounted return

$$J(\pi) \;=\; \mathbb{E}_{\tau \sim p^{\pi}_{\mathcal{M}}}\left[\sum_{t=0}^{T} \gamma^t R(x_t, a_t)\right]. \tag{1}$$

Latent world model. To enable optimization of π through imagined rollouts rather than environment interaction, we equip the agent with a latent world model that compresses past experience and predicts future trajectories. Two encoders together form the *context encoder*,

$$z_t \sim e_\phi(\cdot \mid x_t), \qquad h_t \;=\; c_\phi(z_{\le t},\, a_{<t}), \tag{2}$$

mapping each state to a per-step latent token z_t and each history to a latent context h_t (we collectively denote their parameters by ϕ). Given h_t and a candidate action sequence $a_{t:t+H-1}$ of horizon H, the world model defines a conditional generative distribution over future latents and rewards,

$$p_\theta\big(z_{t+1:t+H},\; r_{t:t+H-1} \;\big|\; h_t,\; a_{t:t+H-1}\big), \tag{3}$$

whose parameters θ constitute the *dynamics adapter*.

Diffusion policy. To represent multimodal action distributions that unimodal parametric families cannot capture, we model the policy $\pi_\psi(a_t \mid h_t)$ as a conditional diffusion (or flow-matching) generator over actions (7; 8), conditioned on the latent context h_t produced by the world model. Such policies are highly expressive but, because they implement a conditional generative process, are acutely sensitive to perturbations in their conditioning input h_t—a property that is central to the challenge addressed in this work.

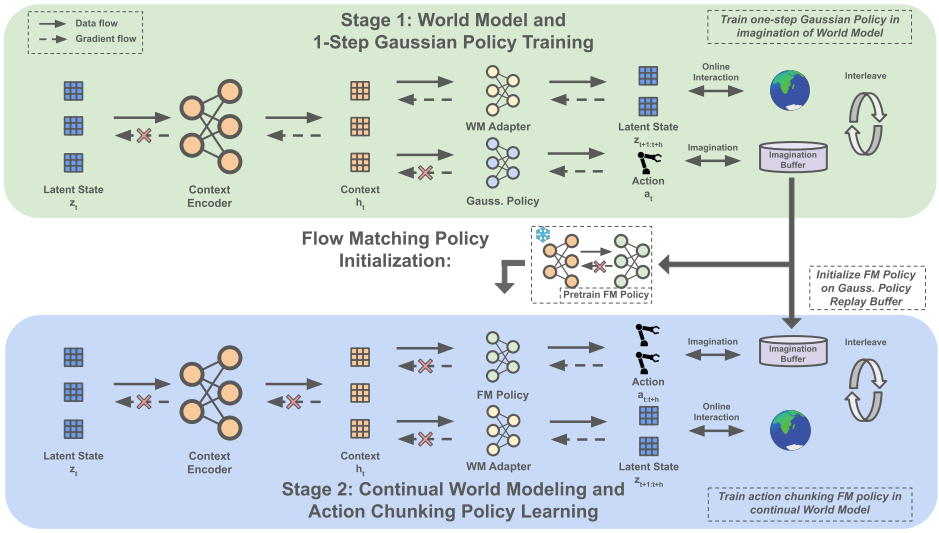


Figure 3: Overview of 2-staged training framework. The first stage trains the world model along with a 1-step Gaussian policy. The second stage begins by pretraining a flow matching policy on the replay buffer of the Gaussian policy, then further training the flow matching policy and world model in an interleaved manner.

MeanFlow parameterization of p_θ. We instantiate p_θ with MeanFlow (42; 43), a few-step variant of flow matching (44) designed for efficient generation. Let $z^1 = z_{t+1:t+H}$ denote the target future latent trajectory and let $z^0 \sim p_0$ be a noise sample from a simple prior. For interpolation time $\sigma \in [0, 1]$, define the linear interpolant and its instantaneous velocity

$$z^\sigma \;=\; (1-\sigma)\, z^0 + \sigma\, z^1, \qquad v(z^\sigma, \sigma) \;=\; z^1 - z^0.$$

The dynamics adapter learns a *mean-velocity* field

$$u_\theta\big(z^\sigma,\, \sigma,\, s \,\big|\, h_t,\, a_{t:t+H-1}\big),$$

where $s \in [0, \sigma]$ is the MeanFlow target time. Following (42), the regression target is

$$u_{\text{tgt}} \;=\; v(z^\sigma, \sigma) \;-\; (\sigma - s)\Big(\, v(z^\sigma, \sigma)\, \partial_z u_\theta \;+\; \partial_\sigma u_\theta \,\Big),$$

and the world-model loss is

$$\mathcal{L}_{\text{WM}}(\phi, \theta) \;=\; \mathbb{E}\Big[\, \big\| u_\theta(z^\sigma, \sigma, s \mid h_t, a_{t:t+H-1}) \;-\; \text{sg}(u_{\text{tgt}}) \big\|_2^2 \,\Big], \tag{4}$$

where $\text{sg}(\cdot)$ is the stop-gradient operator. At inference, future latents are produced by the closed-form MeanFlow update:

$$z^s \;=\; z^\sigma \;-\; (\sigma - s)\, u_\theta\big(z^\sigma, \sigma, s \mid h_t, a_{t:t+H-1}\big), \tag{5}$$

which avoids numerical ODE integration; we use four sampling steps in all experiments. Reward predictions are produced by the world model adapter and trained jointly with u_θ under $\mathcal{L}_{\text{WM}}$.

Offline-to-online setting. In this work, we study offline-to-online learning of a diffusion policy π_ψ alongside a continually adapted world model. An offline stage fits the world model $(e_\phi, c_\phi, u_\theta)$ on a fixed dataset $\mathcal{D}_{\text{off}}$; an online stage then interleaves environment interaction (which grows an environment buffer $\mathcal{B}_{\text{env}}$), world-model updates on $\mathcal{B}_{\text{env}}$, and optimization of π_ψ on imagined rollouts stored in an imagined buffer $\mathcal{B}_{\text{imag}}$. The objective is to maximize $J(\pi_\psi)$. Doing so pulls the world model and the policy in opposite directions: the world model must keep adapting on $\mathcal{B}_{\text{env}}$ to remain accurate on the policy's evolving state distribution, while π_ψ, conditioned on h_t that the world model itself produces, cannot be trained stably under arbitrary changes to it (Appendix C).

4 DiPA-WM: Diffusion Policy Adaptation with Continual World Modeling

To address this challenge, DiPA-WM consists of two complementary parts: a structural decomposition of $(e_\phi, c_\phi, u_\theta)$ that identifies which parameters can be updated online without destabilizing

Environment	Method	Tasks / Metrics			
		Reward	Driving Score	Route Comp.	Infraction
CARLA	DiWA	466.7	0.62	0.77	0.80
	DiPA-WM	**561.5 ± 102.2**	**0.86 ± 0.02**	**0.93 ± 0.03**	**0.92 ± 0.05**
		Hopper	Walker	Ant	Humanoid
MuJoCo	DiWA	2902.6	3389.0	3511.2	4021.9
	DiPA-WM	**3104.7 ± 65.8**	**3722.7 ± 59.4**	**3995.2 ± 117.2**	**4981.2 ±99.4**
		Complete	Mixed	Partial	N/A
FrankaKitchen	DiWA	0.71	0.66	0.61	N/A
	DiPA-WM	**0.85 ± 0.02**	**0.74±0.03**	**0.73±0.01**	N/A
		Can	Square	Transport	N/A
Robomimic	DiWA	0.91	0.81	0.73	N/A
	DiPA-WM	**0.98±0.01**	**0.98±0.00**	**0.81±0.04**	N/A

Table 1: Comparison between DiWA and DiPA-WM across benchmarks. Results are reported as mean ± 1 standard deviation over 3 random seeds for DiPA-WM.

π_ψ (Section 4.1), and a two-stage training procedure that puts this decomposition into practice (Section 4.2).

4.1 Decoupling context representation from dynamics

Where the instability comes from. Among the world model's parameters, only the encoder parameters ϕ enter the computation of h_t. Any update to ϕ during online training therefore shifts the conditioning distribution against which π_ψ's denoiser was fit, which is the representation drift identified in Section 1. The adapter parameters θ, by contrast, change what the world model predicts about the future, but they do not alter what π_ψ sees as input. The instability is therefore localized to a specific subset of parameters, and avoiding it does not require freezing the entire world model.

Why freezing the encoder is enough. The same split also makes sense functionally. The encoder maps state–action histories $(x_{\leq t}, a_{<t})$ to h_t and primarily captures the *semantics* of the environment: what a given history looks like, in a representation that does not depend on the agent's behavior. The adapter takes $(h_t, a_{t:t+H-1})$ and predicts the resulting latent trajectory, capturing the *policy-conditional dynamics*. Environment semantics are largely policy-independent, so a sufficiently diverse mixture of offline and early on-policy data is enough to fit them; later on-policy exploration mostly probes regions whose semantics are already represented. Dynamics, on the other hand, are intrinsically policy-dependent and must be re-fit as the policy evolves.

Frozen encoder, adaptive dynamics. Together, these two observations motivate the central design choice of DiPA-WM. After an initial training stage that exposes the encoder to a broad mixture of offline and early on-policy data, we freeze ϕ and update only θ for the remainder of training. Freezing ϕ pins the distribution of h_t and removes representation drift at its source; continuing to update θ keeps the world model accurate on the evolving on-policy distribution. We operationalize this design as a two-stage procedure (Figure 3).

4.2 Two-stage training procedure

Stage 1: Joint world-model and Gaussian-policy training. Stage 1 produces a well-calibrated world model and an initial policy whose replay buffer will seed the diffusion policy in Stage 2. We first pretrain $(e_\phi, c_\phi, u_\theta)$ on the offline dataset $\mathcal{D}_{\text{off}}$ by minimizing $\mathcal{L}_{\text{WM}}$ (eq. (4)). We then alternate environment interaction with model-based updates of a Gaussian policy $\pi_\eta(a_t \mid h_t)$ (parameterized as an MLP). Each iteration: 1) collects trajectories with π_η and appends them to the environment buffer $\mathcal{B}_{\text{env}}$; 2) updates (ϕ, θ) on $\mathcal{D}_{\text{off}} \cup \mathcal{B}_{\text{env}}$ by minimizing $\mathcal{L}_{\text{WM}}$; and 3) generates imagined rollouts that are stored in an imagined buffer $\mathcal{B}_{\text{imag}}$ and used to update π_η with a model-based RL objective. Policy gradients are stopped at h_t and never flow into ϕ, so the encoder is shaped only by $\mathcal{L}_{\text{WM}}$. We use a Gaussian policy in this stage for two reasons: 1) its simple parametric form tolerates the gradual

Policy	Setting	Reward	Driving Score	Route Completion	Infraction Score
DPPO	Baseline	486.3	0.64	0.79	0.81
	DiPA-WM	**561.5** $\pm$ **102.2**	**0.86** $\pm$ **0.02**	**0.93** $\pm$ **0.03**	**0.92** $\pm$ **0.05**
FPO	Baseline	502.2	0.68	0.82	0.84
	DiPA-WM	**582.4** $\pm$ **85.9**	**0.87** $\pm$ **0.03**	**0.94** $\pm$ **0.03**	**0.92** $\pm$ **0.04**
DSRL	Baseline	497.7	0.66	0.81	0.82
	DiPA-WM	**577.1** $\pm$ **114.8**	**0.87** $\pm$ **0.03**	**0.93** $\pm$ **0.02**	**0.91** $\pm$ **0.06**

Table 2: Experimental results for CARLA navigation.

Policy	Setting	Hopper	Walker	Ant	Humanoid
DPPO	Baseline	2951.1	3543.1	3893.3	4789.3
	DiPA-WM	**3104.7** $\pm$ **65.8**	**3722.7** $\pm$ **59.4**	**3995.2** $\pm$ **117.2**	**4981.2** $\pm$**99.4**
FPO	Baseline	3378.2	4183.1	4217.6	5021.2
	DiPA-WM	**3402.1** $\pm$ **46.1**	**4219.6** $\pm$ **55.2**	**4331.5** $\pm$ **106.8**	**5166.8** $\pm$ **76.3**
DSRL	Baseline	3299.4	4118.9	4091.2	4986.3
	DiPA-WM	**3374.8** $\pm$ **81.4**	**4194.8** $\pm$ **65.9**	**4287.0** $\pm$ **144.9**	**5121.3** $\pm$ **93.7**

Table 3: Experimental results for MuJoCo tasks.

updates of ϕ without the conditional-target misalignment that destabilizes diffusion policies, and 2) its replay buffer supplies a behavior-cloning target for the diffusion policy in Stage 2.

Stage 2: Diffusion-policy adaptation with frozen encoder. Stage 2 introduces the diffusion policy π_ψ. We freeze the encoder parameters ϕ for the rest of training and warm-start π_ψ by behavior cloning on the environment buffer:

$$\psi \leftarrow \arg\min_{\psi} \mathcal{L}_{\mathrm{IL}}(\psi; \mathcal{B}_{\mathrm{env}}). \tag{6}$$

We then alternate at each iteration: 1) collect trajectories with π_ψ and append them to $\mathcal{B}_{\mathrm{env}}$; 2) update only the dynamics adapter on the new data, $\theta \leftarrow \arg\min_\theta \mathcal{L}_{\mathrm{WM}}$, with ϕ held fixed; and 3) generate imagined rollouts with the world model, store them in $\mathcal{B}_{\mathrm{imag}}$, and update π_ψ using the chosen diffusion-policy optimizer (e.g., DPPO, FPO, or DSRL). Because ϕ is fixed, h_t is drawn from a stationary distribution and π_ψ is therefore optimized against a consistent conditional generative target; because θ continues to be updated, the world model remains accurate as the on-policy distribution shifts. This decoupled update schedule is the core of DiPA-WM: it enables simultaneous improvement of the policy and the world model without the instability that frustrates naive joint training.

Architecture and instantiation. All components (e_ϕ, c_ϕ, u_θ, π_ψ) are realized with spatio-temporal transformer backbones; full architectures and hyperparameters are deferred to the appendix. DiPA-WM is independent of the specific diffusion-policy optimizer used in Stage 2, and we report results with three representative choices (DPPO (14), FPO (15), and DSRL (16)) to demonstrate the modularity of the framework.

4.3 Nested Two-Loop Adaptation of World Model and Diffusion Policy

The above design for DiPA-WM is to alleviate the distribution shift that arises when the context encoder is updated online. However, with a frozen context encoder, the model becomes susceptible to non-stationary in the environment over time. To address this challenge, we investigate three strategies to update the context encoder in an online manner.

Hard update. We follow standard stage 2 training, where we alternate between updating the dynamics adapter and the diffusion policy. However, every M training iterations, we perform a full world model update rather than just updating the context encoder. We vary the value of M to ablate the rate at which the context encoder is updated. An overview is provided in Figure 2.

Soft update. We update the full world model every training step, but we scale down the learning rate of the context encoder relative to the rate of the dynamics adapter and diffusion policy, slowing down the training of the context encoder relative to these other components. We vary the learning rate scale in our ablations.

Policy	Setting	Complete	Mixed	Partial
DPPO	Baseline	0.84	0.72	0.68
	DiPA-WM	**0.85 ± 0.02**	**0.74±0.03**	**0.73±0.07**
FPO	Baseline	0.94	0.81	0.86
	DiPA-WM	**0.95 ± 0.02**	**0.83 ± 0.03**	**0.88 ± 0.04**
DSRL	Baseline	0.92	0.80	0.84
	DiPA-WM	**0.95 ± 0.03**	**0.82 ± 0.04**	**0.84 ± 0.06**

Table 4: Results for FrankaKitchen experiments.

Policy	Setting	PickPlaceCan	NutAssemblySquare	TwoArmTransport
DPPO	Baseline	0.84	0.85	0.68
	DiPA-WM	**0.89 ± 0.02**	**0.87 ± 0.03**	**0.74 ± 0.04**
FPO	Baseline	0.98	0.99	0.82
	DiPA-WM	**0.99 ± 0.01**	**0.99 ± 0.01**	**0.86 ± 0.03**
DSRL	Baseline	0.96	0.97	0.72
	DiPA-WM	**0.98 ± 0.01**	**0.98 ± 0.01**	**0.81 ± 0.06**

Table 5: Results for Robomimic experiments.

EMA update. We update the full world model in the every training step, but we additionally maintain an EMA copy of the context encoder which is used to condition the diffusion policy. The EMA copy is used for conditioning the diffusion policy.

5 EXPERIMENTS

We evaluate DiPA-WM across a diverse set of simulated control and robotics environments. We evaluate against the MuJoCo tasks in OpenAI Gym, the Franka Kitchen suite, robotic manipulation in Robomimic, and urban navigation in the CARLA simulator. Additional details are provided in the appendix. Our evaluation is designed to address three key questions:

- Can diffusion policies be trained effectively within a learned world model through iterative online interaction?
- How does our method compare to existing model-based and diffusion-based policy learning approaches?
- How does performance compare to a world model trained entirely offline?

Due to vast architectural differences between various world modeling approaches, we evaluate against other diffusion-policy based world model works. Specifically, we evaluate against DiWA (40), which leverages a pretrained world model to train a diffusion policy in imagination.

Further results, such as world model prediction quality and distribution shift measurements, can be found in Appendix C.

Comparison against offline world modeling To evaluate the benefit of adapting the world model with on-policy data, we compare DiPA-WM to DiWA, which employs a pretrained and frozen world model. Both methods use DPPO for policy optimization. As shown in Table 1, incorporating on-policy updates to the world model leads to significant performance improvements across all tasks.

These gains are particularly pronounced in tasks requiring complex or compositional behaviors, such as the FrankaKitchen and CARLA benchmarks, where offline datasets may not provide sufficient coverage of the state-action space. While both methods optimize their policies using on-policy data, DiPA-WM additionally updates the world model to reflect the evolving data distribution. In contrast, DiWA relies on a fixed model, which can introduce inaccuracies as the policy explores new regions. These results highlight that freezing the world model limits performance under policy-induced distribution shift, and demonstrate the advantage of continually improving both the model and the policy.

LR Scale	Return ($\pm$ std dev)
1.0	1591.2 ± 106.5
0.5	2123.8 ± 142.6
0.33	4247.6 ± 138.5
0.2	4846.4 ± 102.1
0.1	5301.7 ± 97.4
0.05	5321.1 ± 72.8
0.00	5172.9 ± 59.4
EMA (τ =0.95)	5311.4 ± 71.9

Table 6: Return for various context encoder LR scales with soft update rule on the Humanoid task over 3 seeds.

M	Return ($\pm$ std dev)
1	1591.2 ± 106.5
3	4572.1 ± 91.9
5	5191.3 ± 72.8
10	5298.5 ± 68.0
15	5307.6 ± 68.1
20	5306.8 ± 66.3

Table 7: Return for various context encoder training ratios with the hard update rule on the Humanoid task over 3 seeds.

Comparison against model-free approaches Across robotic benchmarks including MuJoCo, FrankaKitchen, and RoboMimic, DiPA-WM enables stable training of flow-matching policies within latent world models and consistently outperforms model-free baselines, as seen in Tables 3, 4, and 5.

These gains are especially apparent in the CARLA navigation task (Table 2), where our approach significantly exceeds the performance of model-free methods. We attribute this improvement to the increased complexity and stochasticity of the environment, which requires long-horizon reasoning and interaction with other agents — capabilities that benefit from planning in a learned world model.

Nested Two-Loop Adaptation of World Model and Diffusion Policy Results for nested two-loop adaptation algorithm in Humanoid and TwoArmTransport are reported in Table 7. In both the hard update and soft update rules, when the context encoder is updated too quickly, performance of the diffusion policy degrades substantially. Performance does not recover to DiPA-WM levels until the value for M is approximately 10, or the learning rate scale is set to 0.10. However, we also note that if the context encoder is allowed to update at a slow enough rate, then final performance improves further. This indicates that allowing the context encoder to adapt to the new state-action distribution is beneficial, but must be controlled. We also note that the EMA update does not provide a meaningful difference in performance, but has additional computational and memory overhead. Additional results can be found in Appendix C.

6 Discussion and conclusion

We presented DiPA-WM, a training framework for diffusion policies in latent world models that supports continual adaptation using on-policy data. We identified a fundamental challenge in this setting: diffusion policies are highly sensitive to changes in their conditioning inputs, while effective model-based reinforcement learning requires updating the world model as the policy evolves. This creates a tension between stable policy optimization and adaptive dynamics learning.

To address this, we proposed a simple yet effective parameterization of the world model into a shared context encoder and a trainable adapter. By freezing the context encoder and continuing to update only the adapter, our approach maintains a stable conditioning space for the diffusion policy while allowing the world model to adapt to new data. Combined with a two-stage training procedure, this enables stable diffusion policy learning alongside continual world model improvement.

Empirically, we demonstrated that DiPA-WM achieves consistent performance gains over both offline world model methods and model-free baselines across a range of benchmarks, with particularly strong improvements in complex environments such as CARLA and FrankaKitchen. Results are further improved when the context encoder is updated online through a nested two-loop approach. These results highlight the importance of continual world model adaptation when training expressive generative policies.

One limitation of our approach is that freezing the encoder constrains the latent representation to remain close to the distribution induced by the offline dataset and early online interaction. If the policy later encounters substantially different observations or behaviors, the fixed representation may become suboptimal and limit the expressivity of the world model. An important direction for future work is therefore to develop mechanisms for continual representation adaptation that preserve stable conditioning for diffusion policies, such as representation regularization techniques.

REPRODUCIBILITY STATEMENT

The appendix specifies the notation, network architecture, learning schedules, hyperparameters, task definitions, and evaluation procedures. It also reports training curves, reward prediction error, state prediction error, continue prediction error, and representation drift measurements.

7 AI USE STATEMENT

In this work, we did not use generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, interpreting results. Additionally, we used generative AI tools for modifying figures, editing code, and polishing the paper. We have carefully reviewed all AI-assisted work, including checking summarized related work for potential hallucinations and verifying AI-generated code for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

[1] D. R. Ha and J. Schmidhuber, "World models," *ArXiv*, vol. abs/1803.10122, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4807711

[2] D. Hafner, J. Pavukonis, J. Ba, and T. P. Lillicrap, "Mastering diverse domains through world models," *ArXiv*, vol. abs/2301.04104, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:255569874

[3] T. Pearce, T. Rashid, D. Bignell, R. Georgescu, S. Devlin, and K. Hofmann, "Scaling laws for pre-training agents and world models," 2024. [Online]. Available: https://arxiv.org/abs/2411.04434

[4] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[5] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?" 2023. [Online]. Available: https://arxiv.org/abs/2211.15657

[6] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "π_0: A vision-language-action flow model for general robot control," 2026. [Online]. Available: https://arxiv.org/abs/2410.24164

[7] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2256–2265. [Online]. Available: https://proceedings.mlr.press/v37/sohl-dickstein15.html

[8] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *arXiv preprint arxiv:2006.11239*, 2020.

[9] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel, "Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer." 2025. [Online]. Available: https://github.com/google-deepmind/mujoco_playground

[10] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[11] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning," *Conference on Robot Learning (CoRL)*, 2019.

[12] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," 2021. [Online]. Available: https://arxiv.org/abs/2108.03298

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[14] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, "Diffusion policy policy optimization," in *arXiv preprint arXiv:2409.00588*, 2024.

[15] D. McAllister, S. Ge, B. Yi, C. M. Kim, E. Weber, H. Choi, H. Feng, and A. Kanazawa, "Flow matching policy gradients," 2025. [Online]. Available: https://arxiv.org/abs/2507.21053

[16] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, "Steering your diffusion policy with latent space reinforcement learning," *Conference on Robot Learning (CoRL)*, 2025.

[17] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *SIGART Bull.*, vol. 2, no. 4, p. 160–163, Jul. 1991. [Online]. Available: https://doi.org/10.1145/122344.122377

[18] M. P. Deisenroth and C. E. Rasmussen, "Pilco: a model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, 2011, p. 465–472.

[19] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[20] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering atari with discrete world models," *arXiv preprint arXiv:2010.02193*, 2020.

[21] C. Chen, Y.-F. Wu, J. Yoon, and S. Ahn, "Transdreamer: Reinforcement learning with transformer world models," *arXiv preprint arXiv:2202.09481*, 2022.

[22] J. Robine, M. Höftmann, T. Uelwer, and S. Harmeling, "Transformer-based world models are happy with 100k interactions," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=TdBaDGCpjly

[23] V. Micheli, E. Alonso, and F. Fleuret, "Transformers are sample-efficient world models," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=vhFu1Acb0xb

[24] W. Zhang, G. Wang, J. Sun, Y. Yuan, and G. Huang, "Storm: Efficient stochastic transformer based world models for reinforcement learning," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 27 147–27 166. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/5647763d4245b23e6a1cb0a8947b38c9-Paper-Conference.pdf

[25] D. Hafner, W. Yan, and T. Lillicrap, "Training agents inside of scalable world models," 2025. [Online]. Available: https://arxiv.org/abs/2509.24527

[26] L. Zhang, Y. Xiong, Z. Yang, S. Casas, R. Hu, and R. Urtasun, "Copilot4d: Learning unsupervised world models for autonomous driving via discrete diffusion," 2024. [Online]. Available: https://arxiv.org/abs/2311.01017

[27] Y. Zhang, S. Gong, K. Xiong, X. Ye, X. Li, X. Tan, F. Wang, J. Huang, H. Wu, and H. Wang, "Bevworld: A multimodal world simulator for autonomous driving via scene-level bev latents," 2025. [Online]. Available: https://arxiv.org/abs/2407.05679

[28] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*, 2022.

[29] K. Lee, H. Liu, M. Ryu, O. Watkins, Y. Du, C. Boutilier, P. Abbeel, M. Ghavamzadeh, and S. S. Gu, "Aligning text-to-image models using human feedback," *ArXiv*, vol. abs/2302.12192, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257102772

[30] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine, "Training diffusion models with reinforcement learning," 2023.

[31] J. Liu, G. Liu, J. Liang, Y. Li, J. Liu, X. Wang, P. Wan, D. Zhang, and W. Ouyang, "Flow-grpo: Training flow matching models via online rl," *arXiv preprint arXiv:2505.05470*, 2025.

[32] J. Liu, G. Liu, J. Liang, Z. Yuan, X. Liu, M. Zheng, X. Wu, Q. Wang, W. Qin, M. Xia *et al.*, "Improving video generation with human feedback," *arXiv preprint arXiv:2501.13918*, 2025.

[33] C.-X. Gao, C. Wu, M. Cao, C. Xiao, Y. Yu, and Z. Zhang, "Behavior-regularized diffusion policy optimization for offline reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2502.04778

[34] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine, "Idql: Implicit q-learning as an actor-critic method with diffusion policies," 2023. [Online]. Available: https://arxiv.org/abs/2304.10573

[35] T. Zhang, C. Yu, S. Su, and Y. Wang, "Reinflow: Fine-tuning flow matching policy with online reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2505.22094

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[37] M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma, "Learning a diffusion model policy from rewards via q-score matching," in *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[38] I. Nematollahi, B. DeMoss, A. L. Chandra, N. Hawes, W. Burgard, and I. Posner, "Lumos: Language-conditioned imitation learning with world models," 2025. [Online]. Available: https://arxiv.org/abs/2503.10370

[39] Z. Hu, X. Yao, Y. Meng, Z. Bing, and A. Knoll, "Dreaming the unseen: World model-regularized diffusion policy for out-of-distribution robustness," 2026. [Online]. Available: https://arxiv.org/abs/2603.21017

[40] A. L. Chandra, I. Nematollahi, C. Huang, T. Welschehold, W. Burgard, and A. Valada, "Diwa: Diffusion policy adaptation with world models," *Conference on Robot Learning (CoRL)*, 2025.

[41] Z. Jiang, K. Liu, Y. Qin, S. Tian, Y. Zheng, M. Zhou, C. Yu, H. Li, and D. Zhao, "World4rl: Diffusion world models for policy refinement with reinforcement learning for robotic manipulation," 2026. [Online]. Available: https://arxiv.org/abs/2509.19080

[42] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He, "Mean flows for one-step generative modeling," 2025. [Online]. Available: https://arxiv.org/abs/2505.13447

[43] Z. Geng, Y. Lu, Z. Wu, E. Shechtman, J. Z. Kolter, and K. He, "Improved mean flows: On the challenges of fastforward generative models," 2025. [Online]. Available: https://arxiv.org/abs/2512.02012

[44] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," 2023. [Online]. Available: https://arxiv.org/abs/2210.02747

[45] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[46] W. Peebles and S. Xie, "Scalable diffusion models with transformers," *arXiv preprint arXiv:2212.09748*, 2022.

[47] Y. Zhu, "Meanflow: Pytorch implementation," https://github.com/zhuyu-cs/MeanFlow, 2025, pyTorch implementation of Mean Flows for One-step Generative Modeling.

[48] D. Gao, S. Cai, H. Zhou, H. Wang, I. Soltani, and J. Zhang, "Cardreamer: Open-source learning platform for world model based autonomous driving," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

APPENDIX

A DIPA-WM ALGORITHM

Algorithm 1 DiPA-WM: Diffusion Policy Adaptation with Continual World Modeling

Require: Offline data $\mathcal{D}_{\text{off}}$, environment $\mathcal{M}$
Require: Environment buffer $\mathcal{B}_{\text{env}}$, imagined buffer $\mathcal{B}_{\text{imag}}$
Require: Encoders (e_ϕ, c_ϕ), world model adapter u_θ, policies π_η, π_ψ
Require: World model loss $\mathcal{L}_{\text{WM}}$, diffusion loss $\mathcal{L}_{\text{diff}}$, policy update routine PolicyUpdate

Initialize $\mathcal{B}_{\text{env}} \leftarrow \mathcal{D}_{\text{off}}$ and $\mathcal{B}_{\text{imag}} \leftarrow \emptyset$
Pretrain $(e_\phi, c_\phi, u_\theta)$ on $\mathcal{B}_{\text{real}}$ using $\mathcal{L}_{\text{WM}}$
Stage 1: Gaussian policy warm-up
for $k = 1, \ldots, K_{\text{MLP}}$ **do**
    Collect $\tau_{\text{env}} \sim \mathcal{M}(\pi_\eta)$ and add to $\mathcal{B}_{\text{env}}$
    Update (ϕ, θ) on $\mathcal{B}_{\text{env}}$ using $\mathcal{L}_{\text{WM}}$
    Generate imagined rollouts $\hat{\tau} \sim u_\theta(\cdot \mid \pi_\eta)$ and add to $\mathcal{B}_{\text{imag}}$
    Update $\pi_\eta \leftarrow \text{PolicyUpdate}(\pi_\eta, \mathcal{B}_{\text{imag}})$
    Stop policy gradients from π_η into (e_ϕ, c_ϕ)
end for
Freeze representation encoders (e_ϕ, c_ϕ)
Initialize π_ψ via imitation learning on $\mathcal{B}_{\text{env}}$ using $\mathcal{L}_{\text{diff}}$
Reset $\mathcal{B}_{\text{imag}} \leftarrow \emptyset$
Stage 2: Diffusion policy adaptation with continual world modeling
for $k = 1, \ldots, K_{\text{DP}}$ **do**
    Collect $\tau_{\text{env}} \sim \mathcal{M}(\pi_\psi)$ and add to $\mathcal{B}_{\text{env}}$
    Update only θ on $\mathcal{B}_{\text{env}}$ using $\mathcal{L}_{\text{WM}}$ ▷ adapter update
    Generate imagined rollouts $\hat{\tau} \sim u_\theta(\cdot \mid \pi_\psi)$ and add to $\mathcal{B}_{\text{imag}}$
    Update $\pi_\psi \leftarrow \text{PolicyUpdate}(\pi_\psi, \mathcal{B}_{\text{imag}})$
end for
return π_ψ, frozen encoders (e_ϕ, c_ϕ), trained world model adapter u_θ

B MODEL ARCHITECTURE

Transformer architectures. All components in DiPA-WM are implemented as stacks of transformer blocks. For image-based environments, latent observations are represented as sequences of spatial tokens over time, and each transformer block factorizes attention into separate spatial (45) and temporal operations following prior spatio-temporal world models (27; 26). Spatial attention is applied independently within each time step, allowing information exchange across spatial patches, while temporal attention is applied independently across time for each spatial location, enabling the model to capture temporal dependencies and dynamics. Each block concludes with a position-wise feedforward network.

For state-based environments, observations are represented as a single token per timestep rather than a spatial grid. In this setting, each model is implemented using standard transformer blocks with full self-attention instead of factorized spatial-temporal attention.

Context encoder. The context encoder consists of a stack of spatio-temporal transformer blocks that processes latent observation tokens together with past action tokens to produce the latent context representation h_t. For image-based tasks, the encoder alternates spatial attention, temporal attention, cross-attention over actions, and feedforward layers. Temporal attention is applied with causal masking to ensure that each latent token attends only to current and past timesteps. The resulting latent tokens are used to form the context representation used by both the world model adapter and diffusion policy.

World model adapter. The world model adapter is implemented as a stack of conditional MeanFlow transformer blocks that predicts future latent trajectories conditioned on the latent context h_t, future

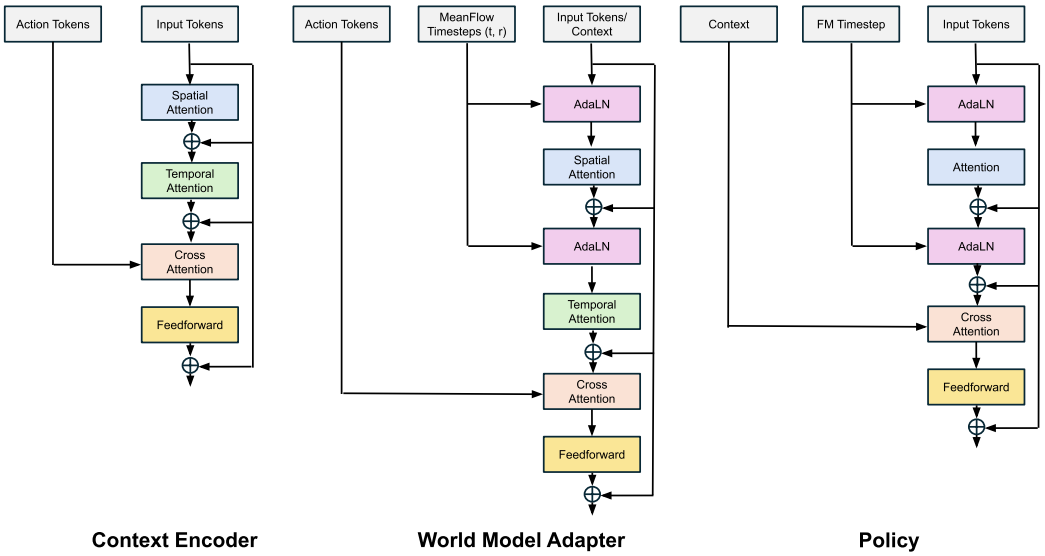


Figure 4: Architecture overview of the context encoder, world model adapter, and diffusion policy used in DiPA-WM. The context encoder processes latent observation and action tokens using factorized spatial-temporal attention to produce the latent context representation h_t. The world model adapter is implemented as a conditional MeanFlow transformer that predicts future latent trajectories conditioned on the latent context, future action tokens, and MeanFlow timesteps (t, r). The diffusion policy is implemented as a conditional flow-matching transformer that generates actions conditioned on the frozen latent context representation and diffusion timestep embeddings. Both the world model adapter and policy share a transformer-based conditional generation architecture, while differing in their conditioning inputs and prediction targets.

action tokens, and MeanFlow timesteps (t, r). The adapter uses adaptive layer normalization (AdaLN) (46) to inject timestep conditioning into the transformer activations prior to attention operations. For image-based tasks, the adapter uses factorized spatial-temporal attention together with cross-attention over future action sequences. We use causal masking in the dynamics adapter to ensure states cannot attend to future states in the input sequence. Additionally, this enables one-step state prediiciton for the gaussian policy during stage 1. We base our MeanFlow implementation on (47).

Diffusion policy architecture. The diffusion policy is implemented as a stack of conditional flow-matching transformer blocks that generates actions conditioned on the latent context representation and diffusion timestep embeddings. Similar to the world model adapter, timestep conditioning is incorporated through AdaLN modulation. The policy additionally includes cross-attention over the latent context representation to condition action generation on the encoded interaction history.

C Further Experiments and Ablations

Setting	Return
One Stage (Joint-Update)	257.2 ± 11.3
One Stage (Frozen)	2296.1 ± 114.8
Two Stages (DiPA-WM)	5172.9 ± 59.4

Table 8: Ablation of the two-stage training design on the Humanoid task over 3 seeds.

Two-Stage Training We next ablate the two-stage training procedure. DiPA-WM exposes the context encoder to online data during stage 1 training with a one-step Gaussian policy. We hypothesize this is an crucial step as attempting to train the context encoder directly with the diffusion policy from scratch would introduce instability, and early freezing would prevent the model from learning

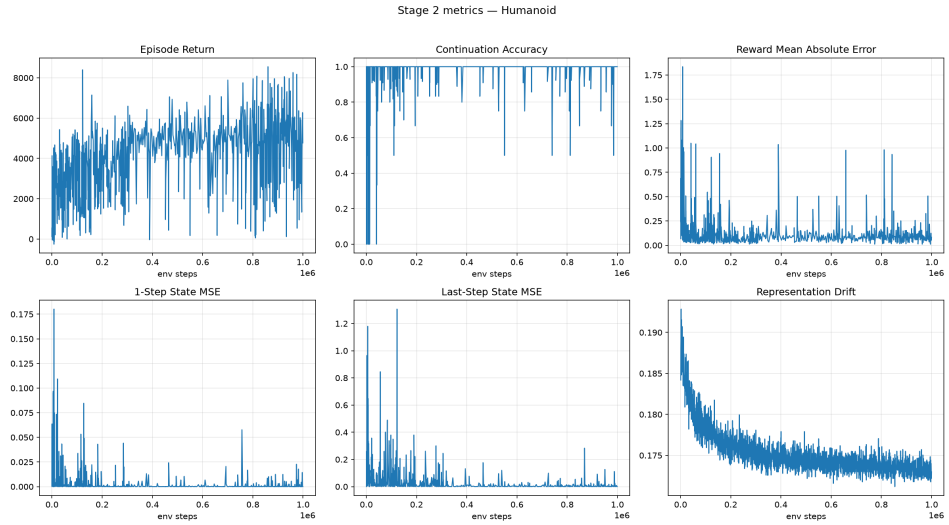


Figure 5: Episode return, continue accuracy, reward mean average error, state MSE, and representation drift evaluated on the Humanoid task with FPO optimizer.

strong representations. We verify our hypothesis by running additional experiments to ablate the two-stage training procedure.

We run experiments on Humanoid to test the impact of training a one-step Gaussian policy followed by the diffusion policy. We run two additional experimental settings to ablate this choice: (1) a single-stage training procedure where the context encoder is updated jointly with the diffusion policy and world model, and (2) a single-stage training procedure where the context encoder is frozen and is trained only from offline pre-training. We compare results to DiPA-WM.

Results are reported in table 8. As shown, when the context encoder is jointly trained along with the world model and diffusion policy directly from scratch, the performance degrades drastically. In the case Humanoid, the policy degenerates to a highly random policy and fails to converge entirely. When the world model and context encoder are pretrained from offline data, and stage 2 training begins directly without stage 1 training, then we see that perorrmance recovers, but still fails to find a strong policy. Although the dynamics adapter is allowed to update, and allowed the world model to correct its prediction errors, the frozen context encoder acts as a bottleneck for both the world model and diffusion policy. In contrast, DiPA-WM leverages two stages of training to expose the context encoder to early online data, before freezing it during stage 2 for stable conditioning for the diffusion policy, and outperforms both ablations.

Metric	Value
1-Step State MSE	0.0004
16-Step State MSE	0.0011
Reward Mean Average Error	0.06
Continue Accuracy	0.998

Table 9: World-model prediction quality on the Humanoid task.

World Model Prediction Quality Finally, we investigate the world model prediction quality throughout training. We are interested in investigating the quality of the state, reward, and continue predictions. We report the one-step state prediction error along with the 16-step prediction error at the end of an imagined rollout. We also report the reward mean average error, as well as the average continue prediction accuracy. We report results in table 9. We see that the world model learns a highly accurate simulator for the MuJoCo Humanoid task.

Distribution Shift Investigation We conduct experiments on the Humanoid task to quantify representation drift in the context encoder during Stage 2. Our primary hypothesis is that updating

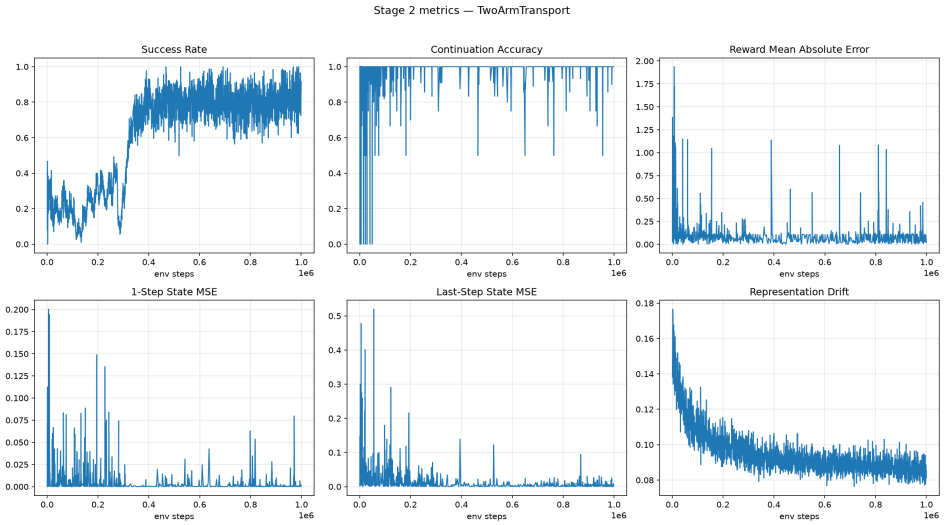


Figure 6: Episode return, continue accuracy, reward mean average error, state MSE, and representation drift evaluated on the TwoArmTransport task with FPO optimizer.

the context encoder online causes shifts in the input representation for the diffusion policy, causing instability. We are interested in measuring the amount of drift experienced by the context encoder. For each training batch, we cache the context vectors produced during the forward pass immediately before a world-model gradient update. After updating the world-model parameters, we recompute the context vectors for the same batch and measure the L_2 distance between the pre- and post-update representations. We normalize this distance by the norm of the original context vectors, yielding the relative representation drift induced by a single world-model update.

When the context encoder is jointly optimized with the world model during Stage 2, we observe substantial and persistent representation drift. The normalized drift is approximately 19% at the beginning of training, decreases to roughly 16%, and subsequently oscillates between 16% and 18.5%. Thus, each world-model update changes the context representations by approximately $\mathbf{17}\%$ on average.

Because the diffusion policy is trained on these context representations, it must continually adapt to a moving conditioning space. This nonstationarity impedes learning of a stable velocity field and leads to substantial performance degradation. Consistent with this interpretation, the joint-update variant achieves a final return of only 1591.2 ± 106.5 on Humanoid.

LR Scale	**Success Rate ($\pm$ std dev)**
1.0	0.21 ± 0.09
0.5	0.38 ± 0.11
0.33	0.63 ± 0.10
0.2	0.74 ± 0.05
0.1	0.88 ± 0.03
0.05	0.90 ± 0.02
0.00	0.86 ± 0.03
EMA (τ =0.95)	0.88 ± 0.02

Table 10: Success rate for various context encoder LR scales on the TwoArmTransport task over 3 seeds.

Further results for nested two-loop adaptation of world model and diffusion policy We perform additional experiments for the nested two-loop adaptation regime on the TwoArmTransport task from Robomimic. Results are reported in table 11 and table 10. We again see that allowing the context encoder of the world model to track the changes in the diffusion policy leads to overall higher performance.

M	Success Rate ($\pm$ std dev)
1	0.21 ± 0.09
3	0.55 ± 0.11
5	0.78 ± 0.06
10	0.88 ± 0.05
15	0.88 ± 0.03
20	0.89 ± 0.02

Table 11: Success rate for various context encoder training ratios with the hard update rule on the TwoArmTransport task over 3 seeds.

Table 12: Hyperparameters used for DiPA-WM.

Component	Hyperparameter	Value
Data and rollout	Environment buffer size $\mathcal{B}_{\text{env}}$	200000
	Action chunk size	4
	Imagined rollouts per policy update	100
	Discount factor γ	0.995
Representation / world model	Number of context encoder layers	3
	Number of world model adapter layers	2
	Transformer hidden dimension	512
	Number of attention heads	4
	Dropout	0.10
MeanFlow world model	MeanFlow sampling steps	4
	World model prediction horizon	4
	Reward prediction loss weight	1.0
	Discount / termination loss weight	0.5
	World model loss weight	1.0
Optimization	Optimizer	AdamW
	Learning rate	1×10^{-4}
	Weight decay	1×10^{-6}
	Batch size	256
	PPO clip ratio	0.01
	GAE parameter λ	0.95

D Experimental Details

Training Environments

OpenAI Gym We first consider four continuous control tasks from the OpenAI Gym benchmarks: Hopper, Walker2d, Ant, and Humanoid. We use the D4RL medium datasets for offline pretraining. Observations consist of low-dimensional state inputs.

Franka Kitchen We evaluate on the Franka Kitchen manipulation environment, where a robotic arm must complete a sequence of tasks (e.g., opening a microwave or moving a kettle). We use three datasets from D4RL: Kitchen-Complete, Kitchen-Partial, and Kitchen-Mixed. We again use state-based inputs.

Robomimic We evaluate on robotic manipulation tasks from the RoboMimic benchmark, which involve learning from human demonstrations in simulated environments. We consider pixel and state inputs.

CARLA navigation Finally, we consider an autonomous driving task in the CARLA simulator. In this environment, an ego vehicle is initialized at a random location in the Town02 map, and a route is generated randomly. 50 background vehicles are spawned and controlled via the CARLA

autopilot. The goal is to navigate to the destination while obeying traffic rules and avoiding collisions. We collect approximately 150GB of demonstration data using the CARLA autopilot for offline pretraining. We use the navigation environment from CarDreamer, an open-source world model framework for autonomous driving tasks (48). The Infraction Score, Route Completion, and Driving Score follow the standard CARLA evaluation protocol. The Infraction Score is initialized at 1.0 and is reduced multiplicatively according to the type and severity of each infraction. Route Completion is the percentage of the designated route completed by the agent. The Driving Score is computed as the product of Route Completion and the Infraction Score. The reward function is based on the reward used in CarDreamer, with additional penalty terms for red-light and stop-sign violations.

Compute Experiments were run on 4 A100 GPUs. In the CARLA navigation task, we pretrain the world model and vision encoder over 62 hours, and performed RL training for another 22 hours. For state-based tasks we perform offline training for 6 hours, followed for RL training for 6 hours.