DiWA: Diffusion Policy Adaptation with World Models

Akshay L Chandra[1*], Iman Nematollahi[1*], Chenguang Huang[2], Tim Welschehold[1], Abhinav Valada[1]
[1]University of Freiburg [2]University of Technology Nuremberg [*]Equal contribution

***Abstract*—Fine-tuning diffusion policies with reinforcement learning (RL) is challenging due to the long denoising sequence, which impedes reward propagation, and the high sample requirements of standard RL. While prior work frames the denoising process as a Markov Decision Process to enable policy updates, it still relies heavily on costly environment interactions. We propose DiWA, a novel framework that fine-tunes diffusion-based robotic skills entirely offline using a world model and RL. Unlike model-free methods that require extensive online interaction, DiWA leverages a world model trained on just a few hours of teleoperated play, enabling efficient and safe adaptation. On the CALVIN benchmark, DiWA improves performance across eight tasks using only offline adaptation, while baselines rely on hundreds of thousands of real-world interaction steps. To our knowledge, this is the first method to fine-tune diffusion policies for real-world robotic skills using an offline world model.**

I. Introduction

Diffusion models have proven effective for robot policy learning, capturing complex multi-modal behaviors through conditional denoising [1, 2]. However, when trained solely via imitation learning, they inherit its core limitations—namely poor generalization under distribution shift and reliance on imperfect demonstrations [3]. Reinforcement learning (RL) addresses these shortcomings by enabling agents to learn from trial and error, improving robustness and generalization [4, 5, 6, 7, 8]. This fine-tuning paradigm, successful in language and vision [9, 10, 11, 12], is particularly appealing in robotics. However, fine-tuning in robotics is hampered by high costs, safety risks, and sample inefficiency of real-world interactions.

Diffusion Policy Policy Optimization (DPPO) [13] adapts diffusion models using Proximal Policy Optimization [14], achieving strong performance in simulation. Yet, it demands millions of environment steps and access to ground-truth simulator states, limiting real-world applicability due to sim-to-real gaps [15] and lack of low-level observations. In contrast, humans leverage internal world models for efficient adaptation. Inspired by this, learned world models [16, 17, 18] offer an appealing alternative to simulators. They enable policy improvement through imagined rollouts, bypassing costly online trials. Recent work [19] shows that world model-trained policies can transfer to the real world without further physical fine-tuning.

We present **DiWA**, the first framework to fine-tune diffusion policies fully offline using a learned world model. DiWA treats the world model as a data-driven simulator, generating latent-space rollouts to fine-tune pre-trained diffusion policies via on-policy RL. This integration of diffusion expressiveness, policy gradient stability, and world model imagination enables safe, efficient robot skill adaptation. In summary, our contributions are threefold:

- **Offline Diffusion Fine-Tuning via World Models:** We introduce DiWA, the first method to fine-tune diffusion policies offline using a learned world model, defining a Dream Diffusion MDP with no real/simulated interaction.
- **Sample-Efficient Adaptation:** Trained on unstructured play data, DiWA refines policies via imagined rollouts, achieving superior sample efficiency on CALVIN.
- **Zero-Shot Real-World Deployment:** We show that diffusion policies fine-tuned entirely within a world model can be deployed on real robots with no additional physical interaction.

II. Problem Formulation

We investigate the problem of offline fine-tuning of diffusion policies for robotic skill adaptation. We assume access to two types of offline datasets: a small set of expert demonstrations $\mathcal{D}_{\text{exp}}$ that are specific to the target skill, and a larger task-agnostic dataset of unstructured play $\mathcal{D}_{\text{play}}$. We model the real environment as a partially observable Markov Decision Process $\mathcal{M}_{\text{env}} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state-observation space, $\mathcal{A}$ the continuous action space, $P(s_{t+1} \mid s_t, a_t)$ the transition dynamics, $R(s_t, a_t)$ the reward function, and $\gamma \in (0, 1)$ the discount factor. A diffusion policy $\pi_\theta(a_t \mid s_t)$ generates actions by first sampling Gaussian noise $\bar{a}_t^K \sim \mathcal{N}(0, I)$, then progressively denoising it through learned transitions:

$$\bar{a}_t^{k-1} \sim \pi_\theta(\bar{a}_t^{k-1} \mid s_t, \bar{a}_t^k), \quad \text{for} \quad k = K, K-1, \ldots, 1, \quad (1)$$

where the final output $\bar{a}_t^0$ is taken as the environment action a_t. The diffusion policy π_θ is first pre-trained via behavior cloning on $\mathcal{D}_{\text{exp}}$, imitating expert actions through denoising. However, behavior cloning is limited by distribution shift and the quality of demonstrations. To address this, we fine-tune the pre-trained policy to maximize expected cumulative reward in the real environment:

$$\theta^\star = \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]. \quad (2)$$

Direct fine-tuning in $\mathcal{M}_{\text{env}}$ is impractical due to high sample complexity and real-world safety concerns. Instead, we train a latent dynamics model on $\mathcal{D}_{\text{play}}$ and define a world model MDP $\mathcal{M}_{\text{wm}} = (\mathcal{Z}, \mathcal{A}, P_\phi, R_\psi, \gamma)$, where $\mathcal{Z}$ is the learned latent space. Fine-tuning is then performed entirely within $\mathcal{M}_{\text{wm}}$, allowing for efficient and safe offline policy adaptation through imagined rollouts.

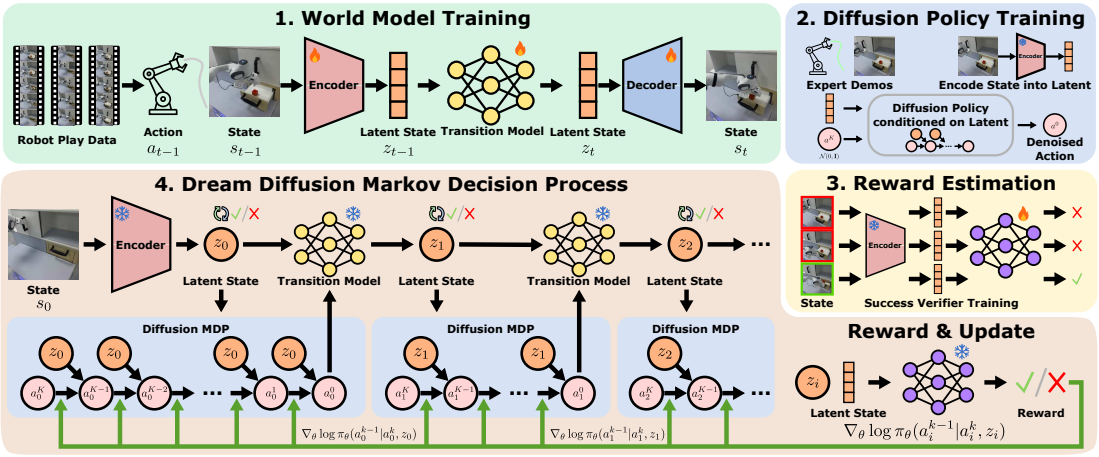


Fig. 1: **DiWA** framework: (1) A world model is trained on robot play data to learn latent dynamics. (2) A diffusion policy is pre-trained on expert demonstrations using latent representations. (3) A success classifier is trained on expert rollouts to estimate task rewards. (4) The diffusion policy is fine-tuned entirely offline via imagined rollouts within the Dream Diffusion MDP, using policy gradients and classifier-based rewards.

III. Offline Adaptation of Diffusion Policy

In this section, we introduce **DiWA**. The training process consists of four phases: (1) learning a world model from an unlabeled play dataset $\mathcal{D}_{\text{play}}$, (2) pretraining a diffusion policy to imitate expert actions from latent representations of $\mathcal{D}_{\text{exp}}$, (3) training a reward classifier on those latents to equip the world model with a task-specific reward, and (4) fine-tuning the policy entirely within the latent space of the world model. At inference time, the fine-tuned policy is deployed in the real environment without any additional adaptation. Figure 1 provides an overview of the approach. For details on hyperparameters and architecture choices, please refer to the appendix.

A. World Model Learning

We train a latent dynamics model on the unlabeled play dataset $\mathcal{D}_{\text{play}}$ to enable offline policy adaptation. The learned world model defines a latent-space MDP $\mathcal{M}_{\text{wm}} = (\mathcal{Z}, \mathcal{A}, P_\phi)$, where $\mathcal{Z}$ is the learned latent space and P_ϕ denotes the transition dynamics. Following prior work [18, 19], we use a recurrent state-space model architecture with an encoder, dynamics model, and decoder. At each timestep t, the model maintains a deterministic recurrent state h_t updated by a transition function f_ϕ, and samples a stochastic latent variable z_t from a posterior conditioned on the current observation x_t:

$$\begin{aligned} &\text{Recurrent state:} && h_t = f_\phi(\hat{s}_{t-1}, a_{t-1}) \\ &\text{Representation model:} && z_t \sim q_\phi(z_t \mid h_t, x_t) \\ &\text{Dynamics predictor:} && \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\ &\text{Decoder:} && \hat{x}_t \sim p_\phi(\hat{x}_t \mid \hat{s}_t), \end{aligned} \tag{3}$$

where the model state is $\hat{s}_t = (h_t, z_t)$. The posterior q_ϕ and prior p_ϕ are modeled as categorical distributions, optimized using straight-through gradient estimators [20]. The model parameters ϕ are trained by minimizing the negative variational evidence lower bound (ELBO). After training, the world model generates imagined trajectories by rolling out latent states from the learned prior $\hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t)$ without additional observations.

B. Pre-training Diffusion Policies

We pre-train the diffusion policy via behavior cloning on expert demonstrations from $\mathcal{D}_{\text{exp}}$. Observations are encoded into latents using the world model, and the policy learns to iteratively denoise random noise into expert actions. This maximizes the likelihood of demonstrated behavior and provides the initialization for offline fine-tuning within the Dream Diffusion MDP.

C. Latent Reward Estimation from Expert Demonstrations

The world model, trained on task-agnostic play data, lacks a reward signal aligned with the target skill. To address this, we train a binary classifier $C_\psi(z_t)$ on latent states extracted from expert demonstrations $\mathcal{D}_{\text{exp}}$. Each observation s_t is encoded into a latent z_t using the world model encoder, and the classifier is trained to predict task success by treating latents from annotated successful frames as positives. During imagined rollouts in $\mathcal{M}_{\text{wm}}$, rewards are computed as $R_\psi(z_t, a_t) := C_\psi(z_{t+1})$, where $C_\psi(z_{t+1}) \in [0, 1]$ reflects the probability of success. This results in an augmented MDP $\mathcal{M}_{\text{wm}} = (\mathcal{Z}, \mathcal{A}, P\phi, R_\psi, \gamma)$ that supports fully offline fine-tuning in imagined trajectories.

D. Dream Diffusion MDP

As observed in prior work [13, 21, 22], a diffusion denoising process can be represented as a multi-step MDP where the likelihood at each step is accessible. We extend this formalism by embedding the diffusion denoising process into the world model MDP, forming the *Dream Diffusion MDP* $\mathcal{M}_{\text{DD}}$. Let $\bar{t}(t, k) = tK + (K - k)$ index the denoising steps across world model timesteps t and denoising steps k, where K is the total number of denoising steps and k decreases lexicographically from K to 1. At index $\bar{t}(t, k)$, the Dream Diffusion MDP defines the state, action, and reward as

$$\begin{aligned} \bar{s}_{\bar{t}(t,k)} &= (z_t, \bar{a}_t^k), \quad \bar{a}_{\bar{t}(t,k)} = \bar{a}_t^{k-1}, \\ \bar{R}_{\bar{t}(t,k)} &= \begin{cases} R_\psi(z_t, \bar{a}_t^0), & \text{if } k = 1, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

Here, $\bar{a}_t^k$ denotes the intermediate action at denoising step k. The transition dynamics are given by

$$\bar{P}(\bar{s}_{\bar{t}+1} \mid \bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}) = \begin{cases} \delta(z_t, \bar{a}_t^{k-1}), & \text{if } k > 1, \\ P_\phi(z_{t+1} \mid z_t, \bar{a}_t^0) \otimes \mathcal{N}(0, I), & \text{if } k = 1, \end{cases}$$

where $\delta(\cdot)$ denotes a Dirac distribution. At denoising steps $k > 1$, the diffusion policy iteratively denoises $\bar{a}_t^k$ into $\bar{a}_t^{k-1}$ while remaining at latent state z_t. When $k = 1$, the final action $\bar{a}_t^0$ is produced, the world model transitions to z_{t+1}, and a new diffusion process begins from fresh noise. Following Eq. (1), the policy at each inner step of the Dream Diffusion MDP is parameterized as a Gaussian:

$$\bar{\pi}_\theta(\bar{a}_t^{k-1} \mid z_t, \bar{a}_t^k) = \mathcal{N}\big(\bar{a}_t^{k-1}; \mu_\theta(z_t, \bar{a}_t^k, k), \sigma_k^2 I\big) \tag{5}$$

where μ_θ is a neural network output. Since each denoising step defines a Gaussian likelihood, the Dream Diffusion MDP admits a well-defined policy gradient objective. Specifically, we optimize

$$\nabla_\theta \bar{\mathcal{J}}(\bar{\pi}_\theta) = \mathbb{E}^{\bar{\pi}_\theta, \bar{P}} \left[\sum_{\bar{t} \geq 0} \nabla_\theta \log \bar{\pi}_\theta(\bar{a}_{\bar{t}} \mid \bar{s}_{\bar{t}})\, \bar{r}(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}) \right] \tag{6}$$

where $\bar{r}(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}) := \sum_{\tau \geq \bar{t}} \gamma^\tau \bar{R}(\bar{s}_\tau, \bar{a}_\tau)$ denotes the return. This objective corresponds to the expected cumulative reward over denoising steps and enables gradient-based fine-tuning of diffusion policies through rollouts in the imagined latent space.

E. Fine-tuning within Dream Diffusion MDP

We fine-tune the diffusion policy in the Dream Diffusion MDP $\mathcal{M}_{\text{DD}}$ using Proximal Policy Optimization (PPO) [14]. Inspired by the two-layer structure of DPPO [13], we adapt PPO to operate entirely within imagined rollouts, alternating between denoising steps and latent transitions. The PPO objective is defined as

$$\begin{aligned} \mathcal{L}_{\text{PPO}} = \mathbb{E}^{\bar{\pi}_{\theta_{\text{old}}}}_{(\bar{s}, \bar{a})} \Big[& \min\big(\rho_\theta(\bar{s}, \bar{a}) \hat{A}(\bar{s}, \bar{a}), \\ & \text{clip}(\rho_\theta(\bar{s}, \bar{a}), 1 - \epsilon, 1 + \epsilon) \hat{A}(\bar{s}, \bar{a})\big) \Big] \end{aligned} \tag{7}$$

where ρ_θ is the importance sampling ratio between the new and old policies. The clipping threshold ϵ constrains the policy update to ensure stability. We estimate the advantage at the denoising step k as

$$\hat{A}(\bar{s}_{\bar{t}(t,k)}, \bar{a}_{\bar{t}(t,k)}) = \gamma_{\text{denoise}}^k \Big(\bar{r}(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}) - \hat{V}(z_t) \Big) \tag{8}$$

where $\gamma_{\text{denoise}} \in (0, 1)$ downweights the contribution of earlier, noisier denoising steps, and $\hat{V}$ estimates the value from the latent state z_t.

To enhance stability and ensure reliable transfer to the real environment, we augment the fine-tuning objective with a behavior cloning (BC) regularization term. Although world models trained on large play datasets capture environment dynamics well, they may still contain subtle errors that the RL agent can exploit. This results in policies that perform well in imagination but fail in the real environment [23]. To address this, we constrain the updated policy to remain close to the pre-trained diffusion policy [24]. The resulting objective is

$$\mathcal{L}_\theta = \mathcal{L}_{\text{PPO}} - \alpha_{\text{BC}}\, \mathbb{E}^{\bar{\pi}_{\theta_{\text{old}}}} \left[\sum_{k=1}^{K} \log \pi_{\theta_{\text{pre}}}(\bar{a}_t^{k-1} \mid z_t, \bar{a}_t^k) \right] \tag{9}$$

where $\pi_{\theta_{\text{pre}}}$ is the frozen pre-trained policy and α_{BC} controls the strength of the regularization.

IV. Experimental Evaluation

We evaluate DiWA for fine-tuning diffusion policies in both simulation and the real-world. Our goals are to: (i) assess whether DiWA can effectively fine-tune policies entirely offline and achieve high task success without additional environment interaction; (ii) analyze the impact of world model fidelity and reward classifier accuracy on adaptation performance; and (iii) evaluate the approach's ability to scale to real-world robotic tasks and transfer zero-shot from imagination to physical execution.

A. Simulation Results

We evaluate our method in environment *D* of the CALVIN simulator [25], which features a 7-DoF Franka Emika Panda robot performing diverse tabletop manipulation tasks. CALVIN offers a teleoperated play dataset that is both broad in coverage and easy to collect, making it ideal for training task-agnostic world models. We train the world model on six hours of play data (∼500,000 transitions) and use a small annotated subset (50 demonstrations per skill) to pre-train individual diffusion policies. Evaluation is conducted on eight tasks from the benchmark.

Evaluation Protocol: We compare DiWA to Diffusion Policy Policy Optimization (DPPO) [13], which fine-tunes diffusion policies via PPO by framing the denoising process as a multi-step MDP. Unlike DPPO, which requires direct interaction with the environment, DiWA performs fine-tuning entirely offline using imagined rollouts in the latent space of a learned world model. For a fair comparison, both methods start from the same pre-trained diffusion policies (one per skill) and use the same latent input: DPPO encodes visual observations with the same encoder used in DiWA (results with raw inputs are in the appendix). A key distinction is reward supervision—DPPO uses ground-truth rewards from the environment, while DiWA

Task	**Base** **Success**	**Offline Fine-Tuning DiWA (Ours)** **Success**	**Online Fine-Tuning (DPPO)** [13] **Env. Steps to Match DiWA**
open-drawer	57.8 ± 3.9	**74.4 ± 1.9**	134k ± 27k
close-drawer	59.1 ± 5.1	**92.0 ± 2.0**	346k ± 28k
move-slider-left	62.2 ± 0.6	**83.3 ± 1.8**	271k ± 29k
move-slider-right	62.6 ± 3.6	**82.8 ± 3.5**	250k ± 9k
turn-on-lightbulb	60.6 ± 3.0	**91.9 ± 1.8**	303k ± 16k
turn-off-lightbulb	35.6 ± 2.0	**77.0 ± 2.0**	327k ± 14k
turn-on-LED	48.4 ± 3.7	**86.2 ± 3.5**	495k ± 46k
turn-off-LED	55.3 ± 4.8	**82.3 ± 6.5**	277k ± 32k
Total Physical Interactions	0		∼2.5M

TABLE I: DiWA fine-tunes diffusion policies offline using imagined rollouts in a learned world model. DPPO requires hundreds of thousands of online interactions to reach similar performance. Results are averaged over three seeds.

relies on a learned classifier trained from a small number of demonstrations, making the task more challenging. We report DiWA's performance after 5 million offline fine-tuning steps and compare it to the number of environment interactions DPPO needs to reach the same performance.

Table I reports the average success rates of pre-trained diffusion policies and their fine-tuned counterparts, averaged over three random seeds. DiWA successfully fine-tunes all evaluated robotic manipulation skills entirely offline, without requiring any additional physical interaction. In contrast, the DPPO baseline typically requires several hundred thousand environment interactions to reach a similar level of performance. Importantly, these interactions involve online exploration, which is often unsafe or impractical in real-world robotic settings. Overall, these results highlight that DiWA enables effective skill adaptation using only offline data, offering a safer and more sample-efficient alternative to model-free approaches.

To evaluate the impact of model components on fine-tuning, we compare three variants: (i) DiWA (Vision WM), trained solely on visual inputs; (ii) DiWA (Hybrid WM + Reward Classifier), which incorporates scene state during training but still uses a learned reward classifier; and (iii) DiWA (Hybrid WM + Latent Decoder), which decodes latents into scene state to compute rewards directly. Figure 2 summarizes the differences. Comparing (i) and (ii), hybrid world models yield faster, more stable fine-tuning, likely due to improved latent dynamics from scene state supervision. Comparing (ii) and (iii), decoding-based rewards further boost performance by enabling more accurate reward estimation. While we use DiWA (Vision WM) as our main variant for real-world compatibility, these results highlight the benefits of richer world models and more precise rewards for fine-tuning.

B. Real-World Results

To evaluate DiWA on real-world robotic skills, we conducted experiments with a Franka Emika Panda robot operating in a tabletop environment containing a cabinet and drawer. We collected a play dataset comprising four hours of teleoperated interaction (∼450,000 transitions) using a VR controller to guide the robot. RGB observations were recorded from both a static and a gripper-mounted camera. We evaluated the model on three representative skills: opening the drawer, closing the drawer, and pushing the cabinet slider to the right. To pre-train the diffusion policies and reward classifiers, we collected 50 expert demonstrations per skill. We trained a generative world model on the offline play dataset and found that it was capable of accurate long-horizon predictions in held-out trajectories. Qualitative rollout examples are provided in the appendix. We then used the trained world model to encode expert demonstrations into latent representations, which were used to pre-train separate diffusion policies and reward classifiers for each skill. Finally, we fine-tuned the pre-trained policies for ∼2 million imagination steps entirely within the latent space of the learned world model.

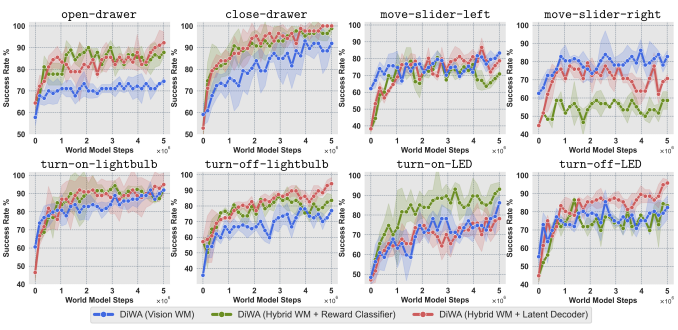


Fig. 2: Comparison of three DiWA variants on simulated fine-tuning tasks. Blue uses only visual inputs, while green and red both incorporate scene state supervision. Red further decodes rewards from latents instead of relying on a learned classifier. Results demonstrate that more expressive world models and more accurate reward signals lead to improved offline fine-tuning performance.

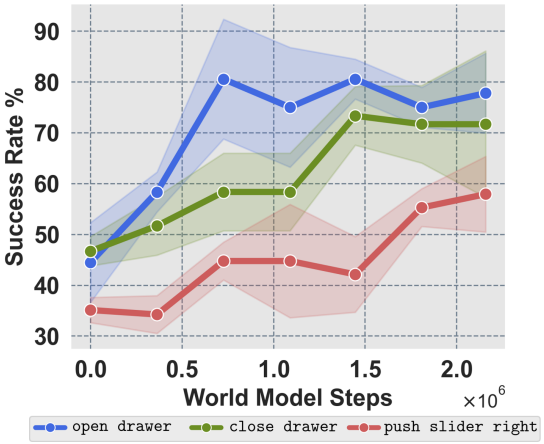


Fig. 3: Success rates before and after offline fine-tuning with DiWA, averaged over 20 rollouts and three seeds. Values correspond to checkpoints saved during fine-tuning. While pre-trained diffusion policies show limited initial performance, DiWA enables significant improvement through imagination-based reinforcement learning without physical interaction.

To evaluate performance, we executed 20 rollouts per skill using fixed initial scene configurations and robot starting positions, both with the pre-trained and fine-tuned policies. Success rates, averaged over three random seeds, are reported in Figure 3. We find that although the pre-trained diffusion policies exhibit limited initial success across all three tasks, DiWA substantially improves their performance through offline fine-tuning within the learned world model. This demonstrates effective real-world policy adaptation without requiring any physical interaction.

V. Conclusion

We presented **DiWA**, a fully offline framework for adapting diffusion policies using learned world models. By treating the world model as a safe, data-driven simulator, DiWA enables reinforcement learning entirely in imagination, avoiding the cost and risk of online interactions. Our approach fine-tunes pre-trained diffusion policies through long-horizon rollouts in latent space, leveraging a compact and expressive representation of environment dynamics. On the CALVIN benchmark, DiWA achieves strong adaptation performance while requiring no additional environment interaction, demonstrating substantial gains in sample efficiency over model-free baselines. Our work provides the first empirical evidence that diffusion policies fine-tuned entirely offline within a learned world model trained on real-world play data can transfer zero-shot to real-world robotic systems.

ACKNOWLEDGEMENTS

This work was supported by the BrainWorlds initiative of the BrainLinks-BrainTools center at the University of Freiburg.

REFERENCES

[1] 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*, page 02783649241273668, 2023.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[3] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[4] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[5] Iman Nematollahi, Erick Rosete-Beas, Adrian Röfer, Tim Welschehold, Abhinav Valada, and Wolfram Burgard. Robot skill adaptation via soft actor-critic gaussian mixture models. In *International Conference on Robotics and Automation (ICRA)*, pages 8651–8657, 2022.

[6] Fabian Schmalstieg, Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces. In *The International Symposium of Robotics Research*, pages 52–66, 2022.

[7] Iman Nematollahi, Kirill Yankov, Wolfram Burgard, and Tim Welschehold. Robot skill generalization via keypoint integrated soft actor-critic gaussian mixture models. In *International Symposium on Experimental Robotics*, pages 168–180, 2023.

[8] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. N^2 m^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments. *IEEE Transactions on Robotics*, 39(5):3601–3619, 2023.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

[10] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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

[12] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.

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

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

[15] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, 2019.

[16] David Ha and Jürgen Schmidhuber. World models. *Neural Information Processing Systems*, 2018.

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

[18] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *International Conference on Learning Representations*, 2021.

[19] Iman Nematollahi, Branton DeMoss, Akshay L Chandra, Nick Hawes, Wolfram Burgard, and Ingmar Posner. Lumos: Language-conditioned imitation learning with world models. In *IEEE International Conference on Robotics and Automation*, 2025.

[20] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[21] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[22] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *International Conference on Machine Learning*, 2024.

[23] Robin Schiewer, Anand Subramoney, and Laurenz Wiskott. Exploring the limits of hierarchical world models in reinforcement learning. *Scientific Reports*, 14(1):26856, 2024.

[24] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal.

Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

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

[26] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[27] Victor Talpaert, Ibrahim Sobh, B Ravi Kiran, Patrick Mannion, Senthil Yogamani, Ahmad El-Sallab, and Patrick Perez. Exploring applications of deep reinforcement learning for real-world autonomous driving systems. *arXiv preprint arXiv:1901.01536*, 2019.

[28] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[29] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 31–36, 2017.

[30] Jonathan Booher, Khashayar Rohanimanesh, Junhong Xu, Vladislav Isenbaev, Ashwin Balakrishna, Ishan Gupta, Wei Liu, and Aleksandr Petiushko. Cimrl: Combining imitation and reinforcement learning for safe autonomous driving. *arXiv preprint arXiv:2406.08878*, 2024.

[31] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Rebecca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7553–7560. IEEE, 2023.

[32] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[33] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *International Conference on Robotics and Automation (ICRA)*, pages 6023–6029, 2019.

[34] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[35] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[36] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[37] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565, 2019.

[38] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

[39] Iman Nematollahi, Erick Rosete-Beas, Seyed Mahdi B. Azad, Raghu Rajan, Frank Hutter, and Wolfram Burgard. T3vip: Transformation-based 3d video prediction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

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

[41] Branton DeMoss, Paul Duckworth, Nick Hawes, and Ingmar Posner. Ditto: Offline imitation learning with world models. *arXiv preprint arXiv:2302.03086*, 2023.

[42] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric A. Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *Robotics: Science and Systems*, 2024.

[43] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710, 2023.

[44] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70, 2024.

[45] Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *Conference on Robot Learning*, volume 229, pages 2323–2339, 2023.

[46] Zhi Hou, Tianyi Zhang, Yuwen Xiong, Hengjun Pu, Chengyang Zhao, Ronglei Tong, Yu Qiao, Jifeng Dai, and Yuntao Chen. Diffusion transformer policy. *arXiv preprint arXiv:2410.15959*, 2024.

[47] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *Robotics: Science and Systems*, 2024.

[48] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence

diffusion. *Advances in Neural Information Processing Systems*, 2024.

[49] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *International Conference on Learning Representations*, 2023.

[50] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *International Conference on Machine Learning*, 2022.

[51] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *International Conference on Learning Representations*, 2023.

[52] Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *International Conference on Learning Representations*, 2024.

[53] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *International Conference on Learning Representations*, 2023.

[54] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[55] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

[56] Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[57] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[58] Richard S. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Neural Information Processing Systems*, 1999.

[59] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 79858–79885, 2023.

[60] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024.

[61] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[62] Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *arXiv preprint arXiv:2411.18704*, 2024.

[63] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[64] 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.

[65] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019.

[66] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

APPENDIX

S.1 Related Work

Reinforcement Learning for Robot Policy Adaptation: Imitation learning (IL) provides a sample-efficient way to train policies but often suffers from covariate shift and compounding errors when encountering out-of-distribution states. In contrast, Reinforcement Learning (RL) enables policy improvement through interaction with the environment, using reward signals to guide behavior. Since the success of deep Q-networks (DQN) on Atari [26], RL has been widely adopted in robotics for tasks ranging from locomotion to manipulation [27, 28, 29]. A common paradigm combines IL and RL, first pre-training a base policy from demonstrations and then fine-tuning it using either online interactions [5, 30, 31, 32, 33] or reward signals extracted from offline data [34, 35]. In this work, DiWA extends this two-stage framework to diffusion policies, enabling fine-tuning of pre-trained policies entirely offline via a learned world model.

Reinforcement Learning with World Models: Due to the high cost and complexity of physical interactions in robotics, world models have emerged as a promising alternative for enabling sample-efficient reinforcement learning. These models [16] are predictive representations of environment dynamics that allow agents to plan and learn through imagined trajectories, reducing the need for real-world interaction. World models have been used for both (i) planning [36, 37, 38, 39] and (ii) model-based rollouts to train policies [17, 18, 40]. However, most existing approaches operate in a closed-loop online setting, where the model is continuously updated using data collected by the learning agent, thereby tightly coupling the world model to the downstream task. An alternative paradigm is to learn general-purpose, task-agnostic world models from unstructured, unlabeled data such as play [19, 41]. These models can be reused across tasks by providing auxiliary reward signals or simulating interactions. DiWA follows this paradigm: it learns a general world model once from offline play data, freezes it, and uses it to fine-tune pre-trained policies entirely offline without any model updates.

Reinforcement Learning for Diffusion-Based Policies: Diffusion-based policies (DPs) [1, 42, 43, 44, 45, 46, 47] have recently achieved strong performance in robotic imitation learning due to their stable training and capacity to model multi-modal behaviors. However, their effectiveness is constrained by the coverage and quality of expert demonstrations. To address this, several approaches have explored extending DPs with trajectory diffusion [48, 49, 50], offline Q-learning [51, 52, 53], online reinforcement learning [22, 54, 55], and residual learning [56]. Policy gradient methods [57, 58], which directly optimize the expected return of a policy, have also been applied to fine-tune diffusion models. This includes recent work on fine-tuning text-to-image diffusion models [59, 60], where the denoising process is treated as a multi-step MDP [13, 21, 22]. Our work builds directly on Diffusion Policy Policy Optimization (DPPO) [13], which first demonstrated how to embed the diffusion denoising process into the environment MDP and apply PPO [14] for fine-tuning in control settings. While DPPO enables effective fine-tuning, it relies on online interactions and ground-truth environment signals. DiWA addresses this limitation by replacing the environment MDP with a learned world model, enabling offline fine-tuning entirely through imagined rollouts.

S.2 Hyperparameters and Training Details

S.2.1 World Model

Following the design introduced in LUMOS [19], we adopt a DreamerV2-style latent dynamics model as the backbone of our world model. While DreamerV2 was originally proposed for Atari game environments [18], our setting focuses on robotic manipulation using raw teleoperated play data. To accommodate this domain shift, we integrate two separate visual encoders for the static and wrist-mounted gripper cameras. Their encoded features are concatenated and fused via a fully-connected layer before being passed to the recurrent state-space model (RSSM). This fusion allows the model to jointly reason over both ego-centric and third-person viewpoints during prediction and imagination. Our world model is trained by minimizing the negative variational Evidence Lower Bound (ELBO):

$$\min_{\phi} \mathbb{E}_{q_\phi}\Bigg[\sum_{t=1}^{T} -\log p_\phi(x_t \mid \hat{s}_t) + \beta\,\mathrm{KL}\big(q_\phi(z_t \mid \hat{s}_t)\,\|\,p_\phi(\hat{z}_t \mid h_t)\big)\Bigg] \tag{10}$$

where $\hat{s}_t = (h_t, z_t)$, and β controls the strength of KL regularization. To stabilize learning, we apply KL balancing to modulate gradient flow between the prior and posterior distributions, following the formulation from Hafner et al. [18]:

$$\mathrm{KL}(q \parallel p) = \delta \underbrace{\mathrm{KL}(q \parallel \mathrm{sg}(p))}_{\text{posterior regularizer}} + (1-\delta) \underbrace{\mathrm{KL}(\mathrm{sg}(q) \parallel p)}_{\text{prior regularizer}}, \tag{11}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. We found KL balancing to be crucial for improving the sharpness and consistency of imagined rollouts, as it accelerates the prior's convergence toward the richer posterior distribution.

The stochastic latent code z_t is modeled using a discrete representation composed of 32 categorical variables with 32 possible classes each. This leads to a sparse 1024-dimensional one-hot vector, which we concatenate with the deterministic hidden state h_t of size 1024, yielding a total latent dimensionality of $k = 2048$. We train all components of the world model jointly using sequences of 50 steps sampled from diverse points in long-horizon play episodes. Due to the scarcity of resets in such data, we reset the recurrent state of the RSSM with a small probability ζ to encourage robustness to initialization and better exploitation of temporal context. All hyperparameters are kept identical across simulation and real-world experiments, except for the KL loss scale β, which is set to 0.3 in simulation and 1.0 in real-world training. To maximize coverage of different scene transitions, we sample training subsequences by selecting random start indices within each episode, ensuring the sampled subsequence remains within

Name	Symbol	Value
Batch size	B	50
Sequence length	L	50
Deterministic latent state dimensions	—	1024
Discrete latent state dimensions	—	32
Discrete latent state classes	—	32
Latent dimensions	k	2048
KL loss scale	β	0.3
KL balancing coefficient	δ	0.8
RSSM reset probability	ζ	0.01
World model learning rate	—	3×10^{-4}
Gradient clipping	—	100
Adam epsilon	ϵ	10^{-5}
Weight decay (decoupled)	—	5×10^{-2}

TABLE S.2: Hyperparameters used for training the world model. All values are shared across simulation and real-world experiments, except KL loss scale β, which is 0.3 for simulation and 1.0 for real-world settings.

episode bounds. This configuration is used consistently across both simulated and real-world settings unless otherwise noted (See Table S.2).

S.2.2 Diffusion Policy

We adopt a denoising diffusion probabilistic model (DDPM) [61] to parameterize our base policy. The diffusion policy is trained to imitate expert trajectories using features produced by our frozen world model encoder. Specifically, we featurize each raw observation with the world model to obtain 2048-dimensional latent vectors, which serve as the input to the policy $\pi_\theta(\cdot \mid z_t)$. This featurization ensures compatibility between the policy's training and inference regimes, as the fine-tuned policy will later be conditioned on imagined future latent states. For each skill, we use $N = 50$ expert demonstration trajectories, randomly selected from task-annotated episodes in the CALVIN simulation [25] and manually collected in the real-world environment. The diffusion model is trained with $K = 20$ denoising steps, and follows a chunked prediction strategy: given an observation horizon of 1 step, it predicts a sequence of $T_p = 4$ future actions, of which the first $T_a = 4$ are executed in the environment. The policy is optimized using a behavior cloning objective over the full denoising trajectory:

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{\mathcal{D}_{\text{exp}}} \left[\sum_{t=1}^{T} \sum_{k=1}^{K} -\log \pi_\theta(a_t^{k-1} \mid z_t, a_t^k) \right], \quad (12)$$

where π_θ predicts denoised actions conditioned on the current latent state z_t and noisy action a_t^k.

The policy model is a multi-layer perceptron (MLP) with three hidden layers of size 512, and we apply exponential moving average (EMA) to the policy weights during training, starting from epoch 20, to enhance stability [62]. All policies are trained for 5000 epochs using the Adam optimizer. We use an initial learning rate of 1×10^{-4}, decayed to 1×10^{-5} using a cosine schedule. We apply a weight decay of 1×10^{-6} and use a batch size of 256. These hyperparameters are kept identical across all CALVIN tasks and our real-world skill evaluations (See Table S.3).

When evaluating the DPPO baseline in the CALVIN simulation environment, we also include a variant that has access to ground-truth state information, which has an observation dimensionality of 51. For the vision-based variant, the input consists of RGB images from both the static and gripper cameras, stacked along the channel dimension, resulting in an input shape of $64 \times 64 \times 6$.

Parameter	Symbol	Value
Common Training Parameters (All Skills)		
Observation Horizon	—	1
Number of Demonstrations	N	50
Planning Horizon	T_p	4
Action Horizon	T_a	4
Training Epochs	—	5000
Diffusion Denoising Steps	K	20
Initial Learning Rate	—	1×10^{-4}
Final Learning Rate	—	1×10^{-5}
Weight Decay	—	1×10^{-6}
MLP Dimensions	—	[512, 512, 512]
EMA Decay	—	0.995
EMA Start Epoch	—	20
EMA Update Frequency	—	10
Batch Size	—	256
Observation Dimensions		
DiWA	—	2048
DPPO (Vision WM Encoder)	—	2048
DPPO (Vision)	—	$64 \times 64 \times 6$
DPPO (State)	—	51

TABLE S.3: Training and model hyperparameters for diffusion policy across all CALVIN and real-world tasks.

S.2.3 Latent Reward Estimator

To learn a task-aligned reward signal, we train a latent reward classifier C_ψ using expert demonstration data $\mathcal{D}_{\text{exp}}$. Each observation x_t is encoded into a latent state z_t via the frozen world model encoder. The classifier comprises two components: a two-layer MLP f_ψ that maps latents to an embedding space, and a subsequent two-layer MLP g_ψ that predicts success or failure based on the embedding.

We jointly optimize the model using a combination of contrastive and classification losses. For the contrastive component, we employ the NT-Xent loss [63], which encourages embeddings of positive pairs to be closer than those of negative pairs. Given a batch of N samples, the NT-Xent loss for a positive pair (i, j) is defined as:

$$\mathcal{L}_{\text{NT-Xent}} = -\log \frac{\exp(\text{sim}(f_\psi(z_i), f_\psi(z_j))/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(f_\psi(z_i), f_\psi(z_k))/\tau)}, \quad (13)$$

where $\text{sim}(\cdot, \cdot)$ denotes the cosine similarity, τ is a temperature parameter, and $\mathbb{1}_{[k \neq i]}$ is an indicator function excluding the anchor sample from the denominator.

In parallel, the classification MLP g_ψ operates on the embeddings to predict success labels, trained using standard cross-entropy loss. The overall training objective combines both terms:

$$\mathcal{L}_{\text{reward}} = \mathcal{L}_{\text{NT-Xent}} + \mathcal{L}_{\text{CE}}. \quad (14)$$

The resulting reward function is defined as $R_\psi(z_t, a_t) := \text{softmax}(g_\psi(f_\psi(z_t)))$, which outputs the predicted probability of success given a latent observation.

Both MLPs use ReLU activations, and the model is trained with the Adam optimizer for 100 epochs. See Table S.4 for the full set of hyperparameters.

Parameter	Value
Embedding MLP Dimensions	[512, 512]
Classification MLP Dimensions	[512, 512]
Activation Function	ReLU
Output Activation	Softmax
Training Epochs	100
Batch Size	32
Learning Rate	1×10^{-6}
Temperature Parameter	0.5
Loss Function	Contrastive + Cross-Entropy
Positive Samples	Annotated success frames
Negative Samples	Distant/unsuccessful frames

TABLE S.4: Hyperparameters used for training the latent reward classifier.

S.2.4 Fine-tuning with DiWA

The full pseudocode for DiWA is shown in Algorithm 1. DiWA fine-tunes a pre-trained diffusion policy π_θ using imagined rollouts from a learned world model M_ϕ and reward classifier C_ψ, forming trajectories in the Dream Diffusion MDP $\mathcal{M}_{\text{DD}}$. At each iteration, imagined transitions are stored in a buffer $\mathcal{D}_{\text{itr}}$, advantages are estimated using Generalized Advantage Estimation (GAE) [64], and PPO-style updates [14] are applied to the policy and value function. GAE is computed at the final denoising step ($k = 1$) for each world model timestep.

$$\hat{A}^{\lambda}_{\bar{t}(t,1)} = \sum_{l=0}^{\infty} (\gamma_{\text{WM}}\lambda)^l \bar{\delta}_{\bar{t}(t+l,1)},$$
$$\text{where} \quad \bar{\delta}_{\bar{t}(t,1)} = \bar{R}_{\bar{t}(t,1)} + \gamma_{\text{WM}} V_\nu(\bar{s}_{\bar{t}(t+1,1)}) - V_\nu(\bar{s}_{\bar{t}(t,1)}). \tag{15}$$

To propagate this signal to earlier denoising steps, we apply a denoising discount to obtain step-specific advantages as $\hat{A}_{\bar{t}(t,k)} = \gamma^k_{\text{denoise}} \hat{A}_{\bar{t}(t,1)}$. The policy is fine-tuned using a behavior-regularized PPO objective that augments the clipped PPO loss with a behavior cloning (BC) regularization term. This regularization encourages proximity to the pre-trained diffusion policy $\pi_{\theta_{\text{pre}}}$, mitigating overfitting to model errors during imagination [23, 24]. The full objective is:

$$\mathcal{L}_\theta = \mathcal{L}_{\text{PPO}} - \alpha_{\text{BC}}\, \mathbb{E}^{\bar{\pi}_{\theta_{\text{old}}}} \left[\sum_{k=1}^{K} \log \pi_{\theta_{\text{pre}}}(\bar{a}_t^{k-1} \mid z_t, \bar{a}_t^k) \right], \tag{16}$$

where α_{BC} controls the regularization strength and $\pi_{\theta_{\text{pre}}}$ remains frozen during fine-tuning. To restrict updates to the last K' denoising steps, we subsample $\mathcal{D}_{\text{itr}}$ to include only entries with $k \leq K'$, keeping the base policy $\pi_{\theta_{\text{pre}}}$ frozen for the initial $K - K'$ steps. The value function V_ν is trained to regress the future discounted sum of latent rewards:

$$\mathcal{L}_\nu = \mathbb{E}_{\mathcal{D}_{\text{itr}}} \left[\left(\sum_{l=0}^{T-t} \gamma^l_{\text{WM}} \bar{R}_{\bar{t}(t+l,1)} - V_\nu(z_t) \right)^2 \right], \tag{17}$$

where V_ν takes as input only the latent state z_t from the $\mathcal{M}_{\text{DD}}$. Table S.5 lists the fine-tuning hyperparameters shared across all skills and experiments for both DiWA and the baseline methods. We set the behavior cloning regularization coefficient $\alpha_{\text{BC}} = 0.05$ for all tasks by default, except for `open-drawer`, `close-drawer`, and `turn-on-LED`, where we observed better performance with values of 0.10, 0.025, and 0.025, respectively.

Parameter	Symbol	Value
Planning Horizon (Environment)	T_p	4
Planning Horizon (Actor)	T_a	4
Denoising Steps	K	20
Fine-tuned Denoising Steps	K'	10
Actor Learning Rate	—	1×10^{-5}
Critic Learning Rate	—	1×10^{-3}
Actor MLP Dimensions	—	[512, 512, 512]
Critic MLP Dimensions	—	[256, 256, 256]
Discount Factor (Env /World Model)	γ_{ENV} / γ_{WM}	0.999
Discount Factor (Diffusion Policy)	γ_{DP}	0.99
GAE Smoothing Parameter	λ	0.95
Behavior Cloning Coefficient (default)	α_{BC}	0.05
Batch Size	—	7500

TABLE S.5: Fine-tuning hyperparameters shared across all skills for DiWA and baseline methods.

S.3 Experimental Setup Details

S.3.1 7-DoF Action Framework

All experiments, both in simulation and in the real world, use a 7-dimensional action space defined as:

$$[\delta x, \delta y, \delta z, \delta\phi, \delta\theta, \delta\psi, \textit{gripperAction}]$$

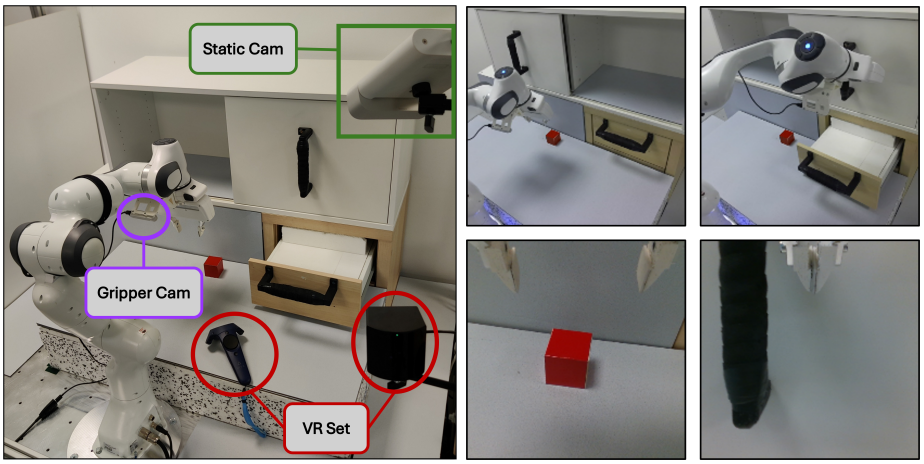
The first six dimensions control the end-effector, with $(\delta x, \delta y, \delta z)$ specifying position changes and $(\delta\phi, \delta\theta, \delta\psi)$ specifying orientation changes via Euler angles. Each takes continuous values in the range $[-1, 1]$. The final dimension, *gripperAction*, controls the gripper state. Although the environment expects discrete inputs (1.0 to close, -1.0 to open), DiWA outputs a continuous value in $[-1.0, 1.0]$, which is thresholded before execution: values greater than or equal to 0 trigger opening, and values less than 0 trigger closing.

S.3.2 Real-World Data Collection

We collected four hours of real-world teleoperation data using a Franka Emika Panda robot controlled via an HTC VIVE Pro headset in a 3D tabletop setting (see Figure S.4a). The tabletop environment included a cabinet with a drawer and a manipulable red cube to support diverse interaction scenarios. During teleoperation, we recorded robot sensor data, including proprioceptive signals (joint states and end-effector pose), as well as multimodal visual observations. RGB images of the full scene were captured at a resolution of 200×200 using an Azure Kinect camera, while close-up RGB views of the manipulated objects were obtained from a wrist-mounted Realsense D415 camera (Figure S.4b). We also logged the absolute control commands sent to the robot. For model training, we computed relative actions as differences between consecutive absolute commands. To reduce redundancy caused by low inter-frame variation, the original 30 Hz recording rate was downsampled by a factor of 4 to 7.5 Hz.

S.4 Data Preprocessing

In both simulation and real-world experiments, we use visual observations from two sources: a static camera and a wrist-mounted gripper camera. All images are first resized to a resolution of 64×64 pixels. We then convert the image tensors from integer values in $[0, 255]$ to floating-point values in $[0.0, 1.0]$, and subsequently normalize them. These transformations are



(a) Real-World Setup (b) Real-World Observations

Fig. S.4: (a) Real-world setup showing the Franka Panda robot, VR teleoperation interface (HTC VIVE controller and tracking system), and camera placements (static Kinect and wrist-mounted Realsense). (b) Example observations from the static and gripper-mounted RGB cameras used during data collection.

applied to both static and gripper observations. In addition to visual observations, we preprocess the robot state, which includes the end-effector's position and orientation. Since the orientation is originally represented in Euler angles, we convert it to a continuous 6D rotation representation [65] to avoid discontinuities and singularities associated with Euler angles.

S.5 Additional Experiments

S.5.1 Comparing DPPO Input Modalities

Figure S.5 compares three DPPO configurations against our offline method. DPPO (State) (gray) uses raw simulator state as input, DPPO (Vision) (red) operates directly on pixel observations using a Vision Transformer (ViT) based encoder [66], and DPPO (Vision WM Encoder) (green) uses visual inputs processed through the same frozen encoder employed in our world model. Among these, the world model latent variant, where DPPO operates on representations produced by our recurrent state space model, often achieves the highest performance, surpassing both raw vision and state-based inputs. These latents combine a history-aware deterministic hidden state with a stochastic component that captures residual uncertainty, providing a compact and dynamics-aligned representation. In contrast to all online variants, DiWA (blue) fine-tunes the policy entirely offline using imagined rollouts in the learned latent space. Its performance is shown as a horizontal band, as no physical interaction is required during fine-tuning. While DPPO can eventually match or exceed our results by leveraging ground truth dynamics and rewards, it requires **hundreds of thousands** of real-world interactions per skill. These interactions are costly, time-consuming, and can pose safety risks. In comparison, DiWA achieves competitive results using only a few hours of play data, offering a safer and more sample-efficient approach to real-world skill adaptation.

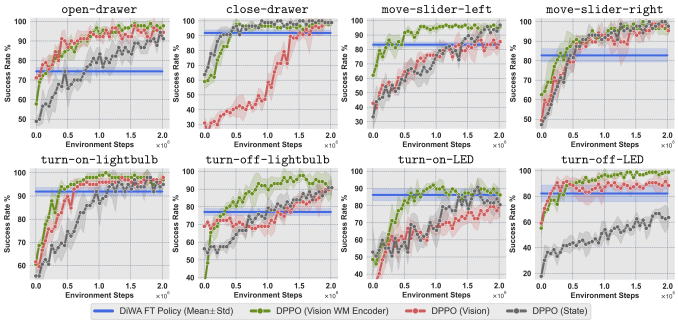


Fig. S.5: Comparison of DiWA with three DPPO variants using different input modalities. DiWA (blue) fine-tunes policies entirely offline using a learned world model, requiring no physical interaction during adaptation. In contrast, DPPO (gray, red, green) performs online reinforcement learning with access to environment rewards and dynamics. The DPPO variant using latents from the world model encoder (green) achieves the highest performance among the three, but all require hundreds of thousands of real-world interactions per skill.

S.5.2 Impact of Behavior Cloning Regularization

To investigate the role of behavior cloning regularization in fine-tuning, we ablate the BC loss coefficient α_{BC} in DiWA and evaluate performance across different settings. As shown in Figure S.6, the choice of α_{BC} has a significant impact on performance.

When $\alpha_{\text{BC}} = 0.0$, meaning no regularization is applied, the agent achieves high success rates during offline evaluation within the imagined environment. However, this performance does not transfer to the real environment, where success rates drop considerably. This discrepancy suggests that the agent overfits to inaccuracies in the world model by exploiting artifacts that yield high imagined rewards but do not correspond to meaningful success in reality [23]. On the other hand, setting α_{BC} too high, such as 0.5, leads to minimal improvement over the pre-trained policy. In this case, strong regularization prevents the policy from effectively adapting to new task-specific feedback, resulting in stagnated learning. Moderate values of α_{BC} provide a better trade-off, enabling

Task	Gaussian Policy	
	Pre-Trained	Offline Fine-Tuned
open-drawer	50.00 ± 0.09	$\mathbf{71.67 \pm 2.36}$
close-drawer	55.17 ± 0.18	$\mathbf{98.28 \pm 2.44}$
move-slider-left	54.86 ± 4.70	$\mathbf{82.64 \pm 1.59}$
move-slider-right	55.52 ± 0.78	$\mathbf{87.93 \pm 7.31}$
turn-on-lightbulb	54.55 ± 3.03	$\mathbf{95.96 \pm 1.75}$
turn-off-lightbulb	62.07 ± 4.88	$\mathbf{77.59 \pm 2.44}$
turn-on-LED	44.83 ± 0.50	$\mathbf{77.59 \pm 7.31}$
turn-off-LED	40.94 ± 3.98	$\mathbf{79.69 \pm 2.21}$
Total Physical Interactions:		**0**

TABLE S.6: Offline fine-tuning improves a unimodal Gaussian policy across all tasks. Success rates increase substantially without any additional real-world interaction.

the policy to adapt while still maintaining alignment with the pre-trained behavior. These results emphasize the importance of tuning BC regularization to balance adaptation and stability when fine-tuning policies with learned world models.

This issue is further compounded by the fact that the world model is trained once on offline play data and remains fixed during fine-tuning. While this avoids the cost and risk of real-world interactions, any modeling errors or artifacts in the learned dynamics persist and may be exploited by the policy. Future work could explore hybrid approaches that incorporate limited online interaction, allowing the world model to be gradually refined with real-world feedback and reducing the impact of such artifacts.

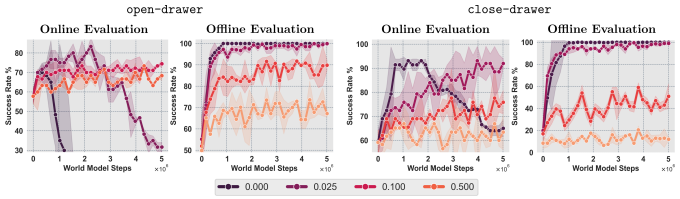


Fig. S.6: Ablation of behavior cloning regularization strength (α_{BC}) during fine-tuning. Without regularization ($\alpha_{\text{BC}} = 0.0$), the agent performs well in imagination but fails in the real environment, indicating exploitation of world model inaccuracies. Excessively high values (e.g., 0.5) prevent meaningful adaptation. Intermediate values strike a balance, yielding robust transfer.

S.5.3 Fine-tuning a Unimodal Gaussian Policy

While the primary focus of this work is on fine-tuning diffusion policies, which involve long denoising sequences that make reward propagation particularly difficult, our method is not limited to this specific policy class. To demonstrate the generality of our formulation, we replace the diffusion policy in DiWA with a unimodal Gaussian policy parameterized by a mean and a diagonal covariance. Unlike diffusion policies, this architecture yields a much shorter Markov chain, allowing reward signals and policy gradients from PPO to propagate more directly. As shown in Table S.6, our fine-tuning procedure leads to consistent improvements across all tasks. This supports the claim that the underlying world model MDP, including the reward estimation mechanism, is independent of the policy architecture.

S.5.4 World Model Rollouts in the Real World

We evaluate the predictive capabilities of our learned world model on real-world hold-out trajectories. As illustrated in Figure S.7, the model generates visually coherent and temporally consistent rollouts over extended horizons. To initiate the prediction, we encode the first two frames of an unseen trajectory to establish the initial context. The model then predicts forward for 80 steps in latent space using its recurrent dynamics, despite being trained with sequences of only 50 steps. The decoded reconstructions from the predicted latents reveal that the world model can accurately track key scene elements, such as the robot arm and manipulated objects, even over long horizons. This highlights the model's ability to learn meaningful dynamics from play data and maintain structured predictions beyond its training horizon.

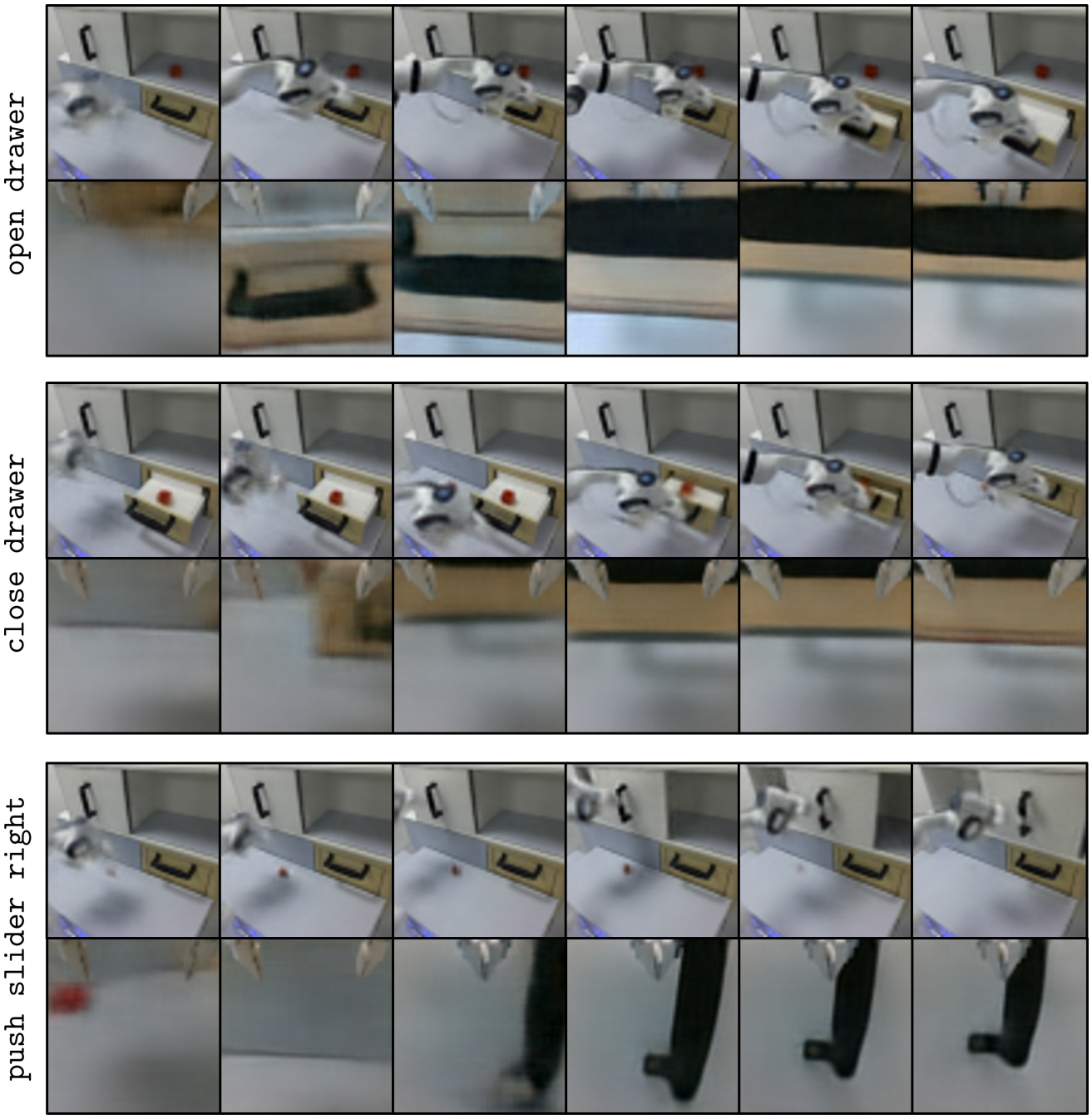


Fig. S.7: Real-world rollout predictions from the learned world model. Each block shows a segment of a held-out trajectory for a specific skill, with static and gripper camera views decoded from imagined latent states. The model produces accurate long-horizon predictions in real-world settings.

Algorithm 1 DiWA: Diffusion Policy Adaptation with World Models

Train world model M_ϕ on play data $\mathcal{D}_{\text{play}}$ using the ELBO objective (Eq. (10)), then freeze M_ϕ.
Encode expert demonstrations into latents $z_t \sim q_\phi(z_t \mid h_t, x_t)$ using the frozen world model.
Pre-train diffusion policy π_θ on latent expert demonstrations via behavior cloning (Eq. (12)); freeze copy as $\pi_{\theta_{\text{pre}}}$.
Train reward classifier C_ψ on latent expert demonstrations via reward loss (Eq. (14)).
Initialize value function V_ν.
for iteration = 1, 2, … **do**
  Initialize imagined rollout buffer $\mathcal{D}_{\text{itr}}$.
  Set $\pi_{\theta_{\text{old}}} = \pi_\theta$.
  for imagination episode = 1, 2, …, N in parallel **do**
    Sample initial observation x_0 and encode to latent z_0.
    Initialize state $\bar{s}_{\bar{t}(0,K)} = (z_0, \bar{a}_0^K)$ in $\mathcal{M}_{\text{DD}}$.
    for imagined step $t = 0, \ldots, T-1$, denoising step $k = K, \ldots, 1$ **do**
      Sample intermediate action $\bar{a}_t^{k-1} \sim \bar{\pi}_{\theta_{\text{old}}}(\cdot \mid z_t, \bar{a}_t^k)$
      if $k = 1$ **then**
        Run final action $\bar{a}_t^0$ in the world model M_ϕ
        Update recurrent state: $h_{t+1} = f_\phi(h_t, \bar{a}_t^0)$
        Sample next latent state: $z_{t+1} \sim p_\phi(z_{t+1} \mid h_{t+1})$
        Predict reward: $\bar{R}_{\bar{t}(t,1)} = R_\psi(z_t, \bar{a}_t^0)$
        Sample new noisy action: $\bar{a}_{t+1}^K \sim \mathcal{N}(0, I)$
        Set next state: $\bar{s}_{\bar{t}(t+1,K)} = (z_{t+1}, \bar{a}_{t+1}^K)$
      else
        Set reward: $\bar{R}_{\bar{t}(t,k)} = 0$
        Set next state: $\bar{s}_{\bar{t}(t,k-1)} = (z_t, \bar{a}_t^{k-1})$
      end if
      Add $(k, \bar{s}_{\bar{t}(t,k)}, \bar{a}_{\bar{t}(t,k)}, \bar{R}_{\bar{t}(t,k)})$ to $\mathcal{D}_{\text{itr}}$.
    end for
  end for
  Compute advantage estimates $A^{\pi_{\theta_{\text{old}}}}(\bar{s}_{\bar{t}(t,1)}, \bar{a}_{\bar{t}(t,1)})$ using GAE (Eq. (15))
  for update = 1, …, num_updates **do**
    for minibatch = 1, …, B **do**
      Sample $(k, \bar{s}_{\bar{t}(t,k)}, \bar{a}_{\bar{t}(t,k)}, \bar{R}_{\bar{t}(t,k)})$ and $A^{\pi_{\theta_{\text{old}}}}(s_{\bar{t}(t,k)}, a_{\bar{t}(t,k)})$ from $\mathcal{D}_{\text{itr}}$.
      Compute denoising-discounted advantage $\hat{A}_{\bar{t}(t,k)} = \gamma_{\text{denoise}}^k A^{\pi_{\theta_{\text{old}}}}(s_{\bar{t}(t,0)}, a_{\bar{t}(t,0)})$.
      Update π_θ using regularized PPO loss (Eq. (16)).
      Update V_ν using value loss (Eq. (17)).
    end for
  end for
end for
return fine-tuned policy π_θ.