SCALING WORLD-MODEL REINFORCEMENT LEARNING THROUGH DIFFUSION POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning (RL) can be effectively supported at scale through the use of world models. However, in practice, scaling such approaches remains fundamentally limited. A commonly recognized challenge is model bias and error compounding, which degrade long-horizon predictions. Beyond these issues, we identify a more critical yet underexplored bottleneck: a structural misalignment between search and value learning in existing world model approaches. In particular, policy improvement often relies on value functions induced by a separate, non-search policy, resulting in training inconsistency and ultimately suboptimal learning. To address this limitation, we propose Model-Based Diffusion Policy Optimization (MBDPO) in world models, a framework that unifies search and policy optimization through diffusion policy representations, thereby unlocking the potential of world models for scalable policy learning. Instead of constructing an explicit planner over a learned world model, we reformulate policy optimization as a diffusion process over searched trajectories in latent world models. In this view, we extract an implicit energy function from the collected dataset that anchors the policy, enabling MBDPO to refine the score field for policy optimization while mitigating misalignment. We evaluate MBDPO across a wide range of settings, including offline pretraining, online learning, offline-to-online fine-tuning, and massively multi-task online learning. MBDPO achieves strong multi-task performance and exhibits consistent scaling behavior with increasing model size.

1 INTRODUCTION

Model-based reinforcement learning (RL) has long been regarded as a promising paradigm for sample-efficient planning in general cases. By learning an explicit dynamical system, agents can dream and plan without multiple interactions with real environments (Kuvayev & Sutton, 1996; Sutton et al., 1998). Recent advances in world models have further expanded this paradigm (Garrido et al., 2024; Zhou et al., 2024; Hafner et al., 2025a; Psenka et al., 2026), enabling large models to capture rich environment dynamics across diverse tasks. Instead of serving solely as local frame predictors (SV et al., 2023; Thrun et al., 1990; Guan et al., 2023), modern world models aim to learn generalizable representations of environment dynamics that can be reused across downstream control problems. At the same time, the emergence of large-scale datasets has created new opportunities for pretraining world models at scale (Wu et al., 2023; Hafner et al., 2025b). Analogous to foundation models in vision and language, pretrained world models offer the potential to unlock scalable model-based RL among multi-tasks (Hansen et al., 2024).

Despite recent advances, model-based RL methods built upon world models still struggle to meaningfully outperform their model-free counterparts (Van Hasselt et al., 2019; Fujimoto et al., 2025; Chang et al., 2026). The most common explanation attributes this shortfall to limited model accuracy and compounding prediction errors (M. Moerland et al., 2023; Lambert et al., 2022). When a learned model is used to simulate long-horizon trajectories, even small one-step prediction errors accumulate over time, resulting in increasingly unreliable state estimates. Motivated by this observation, prior work has focused on mitigating policy degradation induced by imperfect model dynamics. Representative approaches include improving model accuracy (Nagabandi et al., 2018; Hafner et al., 2019), incorporating uncertainty-aware modeling (Deisenroth & Rasmussen, 2011; Curi et al., 2020; Fu et al., 2022), employing truncated or branched rollouts (Janner et al., 2019; Park et al., 2025b), and enhancing long-horizon prediction (Ma et al., 2024; Farebrother et al., 2025). Underlying these

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.

efforts is an implicit assumption: that sufficiently accurate world models will naturally translate into better planning and, ultimately, better control performance. However, recent work challenges this point (Chang et al., 2026), arguing that model accuracy alone does not dictate planning effectiveness. Even highly accurate models can fail to yield strong policies, causing RL within world models to struggle in both performance and scalability. These observations lead us to a central question:

What are the fundamental bottlenecks in scaling model-based RL within latent world models, and how can we architect scalable algorithms to overcome them?

In this work, we show that the core obstacle lies in the misalignment between policy search and value learning, which we refer to as *search-value misalignment*. Due to the substantial computational cost of search, the value function is typically trained using rollouts generated by a learned non-search policy network (Hansen et al., 2022; 2024; 2025; Wang et al., 2025; Zhan et al., 2025; Shimizu & Tomizuka, 2024). In contrast, policy improvement relies on trajectories produced by a search procedure (i.e., model predictive path integral (Williams et al., 2015)) over a learned world model. This mismatch introduces a distributional discrepancy: the value function is trained on state-action pairs generated by a non-search policy, whereas policy execution and improvement are governed by trajectories produced via search over the world model. As a result, the value function is often queried on out-of-distribution state-action pairs where its estimates are unreliable and typically overoptimistic; the search procedure then exploits these errors, leading to systematic overestimation bias and ultimately degrading both performance and scalability (Fujimoto et al., 2019).

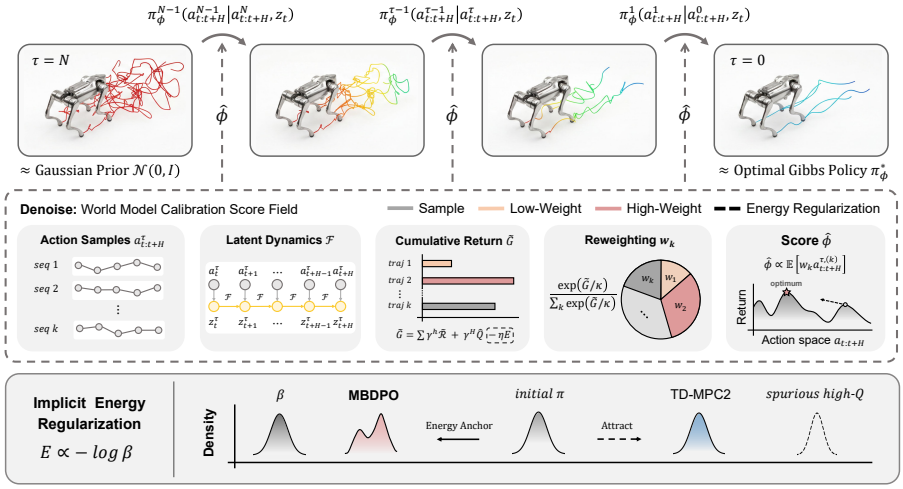

Figure 1: **Core framework of MBDPO.** The world model estimates scores by reweighting action rollouts according to their energy-regularized returns, guiding denoising toward the optimal Gibbs policy. Energy regularization anchors the policy to the non-search policy within a KL trust region, unifying generative denoising, model-based search, and policy optimization.

We address this question with Model-Based Diffusion Policy Optimization (MBDPO). MBDPO reformulates heuristic trajectory search as behavior-policy-anchored diffusion policy optimization inside the learned world model. MBDPO combines an implicit energy learned from collected transitions with imagined world-model rollouts to optimize a diffusion policy while remaining close to the data-supported behavior distribution. By favoring high-return trajectories while limiting policy deviation, this formulation directly addresses the two gaps identified above.

Our contributions are threefold. (1) We identify search–value misalignment as a structural bottleneck in world-model RL and formalize its consequences through the Value Iteration Gap and Policy Improvement Gap. (2) We propose model-based diffusion policy optimization (MBDPO), a unified framework that formulates search and policy optimization as a diffusion process over imagined trajectories of world models, thereby unlocking the potential of world models for scalable policy learning. (3) We show that MBDPO reduces cross-TD error and policy drift, achieves strong performance across a broad single-task suite and the massively multi-task MMBench setting, exhibits favorable model scaling in both offline and online multi-task regimes, and supports efficient offline-to-online adaptation. Additional ablations clarify how behavior anchoring, diffusion refinement, and Monte Carlo score estimation contribute to these results.

2 PRELIMINARIES

Model-Based RL. We consider a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho, \gamma)$ (Puterman, 1990), where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. ρ is the initial state distribution, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function in the state space, and $\gamma \in (0, 1)$ is the discount factor. For a policy $\pi(a|s)$, the action-value function (or Q-function) $Q^\pi(s, a)$ is defined as the expected discounted return starting from state s and action a: $Q^\pi(s, a) \coloneqq \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a\right]$. Our objective is to find a policy π that maximizes the expected discounted return: $\mathbb{E}_{s\sim d^\pi, a\sim\pi}[Q^\pi(s, a)]$. In this paper, we jointly optimize the policy π and a world model that approximates the latent MDP dynamics.

Latent World Model. Drawing on recent advances (Hansen et al., 2024; 2025; Hafner et al., 2025a), we employ a latent world model to capture compact representations essential for complex decision-making tasks. Our world model comprises the following components:

$$\begin{aligned} &\text{Encoder} && \mathcal{E}\colon \mathcal{S} \times \mathcal{E}_{env} \to \mathcal{Z}, && z_t = \mathcal{E}(s_t, e); \\ &\text{Latent Dynamics} && \mathcal{F}\colon \mathcal{Z} \times \mathcal{A} \times \mathcal{E}_{env} \to \mathcal{Z}, && z_{t+1} = \mathcal{F}(z_t, a_t, e); \\ &\text{Latent Reward} && \mathcal{R}\colon \mathcal{Z} \times \mathcal{A} \times \mathcal{E}_{env} \to \mathbb{R}, && \hat{r}_t = \mathcal{R}(z_t, a_t, e); \end{aligned} \tag{1}$$

where $z_t \in \mathcal{Z}$ denotes the latent state and $e \in \mathcal{E}_{env}$ is a learnable task embedding designed for multi-task generalization. For brevity, we *omit* e in subsequent discussions unless its inclusion is explicitly required. Rather than operating in the raw observation space, our RL framework is built entirely upon the learned latent world model.

Diffusion Policy. Instead of generating a single-step action, we optimize an action sequence $a_{t:t+H} \coloneqq \{a_t, \ldots, a_{t+H}\}$ and execute it in a receding-horizon fashion, similar to MPC. We adopt the Denoising Diffusion Probabilistic Model (DDPM) framework (Ho et al., 2020) to represent our generative policy. Let $\tau \in \{1, \ldots, N\}$ denote the diffusion timestep. Since each step of the reverse-time transition is determined by the score function $\phi(z_t, a^\tau_{t:t+H}, \tau) \coloneqq \nabla_{a^\tau_{t:t+H}} \log \pi^\tau_\phi(a^\tau_{t:t+H}|z_t)$, the evolution of the policy is directly governed by ϕ. Accordingly, the reverse diffusion process is characterized by the reverse-time transition kernel $\pi^{\tau-1}_\phi(a^{\tau-1}_{t:t+H}|a^\tau_{t:t+H}, z_t)$, where the marginal distribution of the clean action sequence is

$$\pi^0_\phi(a^0_{t:t+H}|z_t) = \int_{a^{1:N}_{t:t+H}} \pi^N_\phi(a^N_{t:t+H}|z_t) \prod_{\tau=N}^{1} \pi^{\tau-1}_\phi(a^{\tau-1}_{t:t+H}|a^\tau_{t:t+H}, z_t) da^{1:N}_{t:t+H}, \tag{2}$$

which describes a multi-step refinement process that transforms from a Gaussian prior $\pi^N(a^N_{t:t+H}|z_t) \sim \mathcal{N}(0, I)$. Specifically, the stepwise transition kernel is guided by the score function:

$$\pi^{\tau-1}_\phi(a^{\tau-1}_{t:t+H}|a^\tau_{t:t+H}, z_t) = \mathcal{N}\left(a^{\tau-1}_{t:t+H}; \tfrac{1}{\sqrt{\alpha^\tau}}\left(a^\tau_{t:t+H} + (1-\alpha^\tau)\phi(z_t, a^\tau_{t:t+H}, \tau)\right), \sigma^2(\tau) I\right), \tag{3}$$

where α^τ is the scalar factor of noise injection at the τ step, and $\sigma^2(\tau)$ denotes the step-dependent variance. In contrast to conventional diffusion policies in the model-free setting, our approach leverages the world model to estimate the score function ϕ via imagined trajectories, thereby unifying the search and policy optimization. The next section establishes the link between the diffusion refinement in equation 2 and policy optimization within the world model framework.

3 METHOD

In this section, we analyze why existing model-based RL approaches in world models do not correspond to a monotonic policy improvement, and thus fail to guarantee convergence to the optimal policy. To address this issue, we propose a corrected and scalable learning paradigm that reconsiders the interplay between the world model and policy optimization.

3.1 MISALIGNMENT BETWEEN SEARCH AND VALUE LEARNING

We briefly revisit RL strategies built upon world models, such as those proposed in (Hansen et al., 2022; 2025; Zhan et al., 2025; Lin et al., 2025). In these representative approaches, a sampling-based planner operates within the learned world model to generate high-value actions for environment interaction. However, this design inherently introduces a misalignment between policy improvement

and value learning (Chang et al., 2026). Specifically, while the value function is trained on state-action pairs generated by a non-search policy β[1], the executed policy π[2] relies on the sampling-based planner over the world model, resulting in a significant distributional discrepancy (see more details in Appendix F.1.1). This misalignment hinders performance and limits scalability. In particular, we elucidate why such a procedure may fail to guarantee optimality, suggesting that it does not constitute true policy optimization in the conventional sense.

Problem 1 (Value Iteration Gap). Let β denote the learned, non-search policy, and let π be the policy induced by search. Suppose $\hat{Q}$ is the learned value function trained on data collected under β. Then, the discrepancy between the Bellman operators under π and β (denoted as T^π and T^β, respectively; see Definition E.1) introduces the following error when updating $\hat{Q}$:

$$\|T^\pi \hat{Q} - T^\beta \hat{Q}\|_\infty \leq \gamma \|\hat{Q}\|_\infty \sqrt{2 D_{\mathrm{KL}}^{\max}(\pi \| \beta)}, \tag{4}$$

where $D_{\mathrm{KL}}^{\max}(\pi\|\beta) := \sup_{z \in \mathcal{Z}} D_{\mathrm{KL}}(\pi(\cdot|z)\|\beta(\cdot|z))$, see the detailed proof in Appendix F.1.2.

While search aims to improve the policy, it can induce a vicious cycle of error amplification. Since $\hat{Q}$ is trained under β, it is unreliable in out-of-distribution regions. The search policy π tends to select actions that maximize $\hat{Q}$, which often correspond to spurious overestimated values, i.e., $\hat{Q}(z,a) > Q^\beta(z,a)$ for some $(z,a) \in \mathcal{Z} \times \mathcal{A}$. The Bellman update $T^\pi \hat{Q}$ then bootstraps on these erroneous peaks, reinforcing rather than correcting them. As a result, the iterative updates can amplify the estimation error of $\hat{Q}$ as shown in equation 4.

Problem 2 (Policy Improvement Gap). Let $J(\pi)$ and $J(\beta)$ denote the standard discounted returns induced by policies π and β, respectively. Furthermore, let $\hat{J}(\beta) := \mathbb{E}_{z \sim d^\beta, a \sim \beta}[\hat{Q}(z,a)]$ and $\hat{J}_\beta(\pi) := \mathbb{E}_{z \sim d^\beta, a \sim \pi}[\hat{Q}(z,a)]$ represent the estimated Q-based surrogate objectives evaluated under the discounted state occupancy distribution d^β using an approximate value function $\hat{Q}$. Then, the performance gap satisfies the following bound:

$$\underbrace{J(\pi) - J(\beta)}_{\text{true improvement}} \geq \frac{1}{1-\gamma} \underbrace{\left(\hat{J}_\beta(\pi) - \hat{J}(\beta)\right)}_{\text{estimated improvement}} - C_1 D_{\mathrm{KL}}^{\max}(\pi\|\beta) - \underbrace{C_2 \|Q^\beta - \hat{Q}\|_\infty}_{\text{value iteration gap in Problem 1}}, \tag{5}$$

where C_1 and C_2 are positive constants depending on the discount factor γ and the boundedness of the value functions, see the analysis in Appendix F.2.

Equation 5 shows that estimated improvement can be offset by both policy divergence and value approximation error. As π moves away from β, $\hat{Q}$ becomes less reliable for evaluating the searched actions, while the resulting Bellman updates can further amplify the value error identified in Problem 1. Empirically, this phenomenon is characterized by a severe inflation in cross TD error and substantial action drift relative to the base policy, as illustrated in Figure 2. Together, these gaps show that the central bottleneck is not search quality alone, but the separation between search and policy learning. MBDPO therefore recasts trajectory search as behavior-anchored policy optimization inside the world model, making policy improvement structurally consistent with value learning.

3.2 Model-Based Diffusion Policy Optimization (MBDPO) in World Models

In this subsection, we define the policy learning objective and detail the core component of our framework: MBDPO within world models.

Policy Objective. In RL built upon world models, policy optimization is framed as seeking a policy π_ϕ through the following formulation:

$$\arg\max_{\pi_\phi} \mathbb{E}_{a_{t:t+H} \sim \pi_\phi(\cdot|z_t)} \left[G(z_t, a_{t:t+H}) \right], \tag{6}$$

[1]Here, β denotes the data-collection policy associated with the training samples. In offline settings, β is the behavior policy that generated the fixed offline dataset. In online settings, β refers to the implicit replay-buffer policy, i.e., the mixture of historical policies that collected the transitions stored in the replay buffer.

[2]We use π to denote the search policy used in other methods, contrasting it with our diffusion policy π_ϕ.

where

$$G(z_t, a_{t:t+H}) := \sum_{h=t}^{t+H-1} \gamma^{h-t} \mathcal{R}(z_h, a_h) + \gamma^H \hat{Q}(z_{t+H}, a_{t+H}). \tag{7}$$

In prior work (Hansen et al., 2022; 2024; 2025; Zhan et al., 2025; Psenka et al., 2026), the action sequence $a_{t:t+H}$ is typically obtained via model predictive path integral (MPPI). However, this approach can be problematic: MPPI is initialized with a Gaussian prior and relies on trajectories generated by the world model, which can lead to a large KL divergence from the non-search policy β. Consequently, both the Bellman update and policy improvement may enter a detrimental feedback loop, as illustrated in Problems 1 and 2. Essentially, this paradigm does not constitute true policy optimization; it merely performs a heuristic search that lacks the guarantees and distributional consistency required for a monotonic improvement. To address this, it is imperative to establish a principled policy optimization objective that simultaneously maximizes expected returns and constrains the search policy to the non-search base policy β:

$$\begin{aligned} \max_{\pi_\phi} \quad & \mathbb{E}_{a_{t:t+H} \sim \pi_\phi(\cdot|z_t)} \big[G(z_t, a_{t:t+H}) \big] \\ \text{s.t.} \quad & \underbrace{D_{\mathrm{KL}}(\pi_\phi(\cdot|z_t) \| \beta(\cdot|z_t)) \le \varepsilon}_{\text{KL constraint}}, \ \underbrace{z_{h+1} = \mathcal{F}(z_h, a_h), \quad \forall h \in \{t, \dots, t+H-1\}}_{\text{latent dynamics constraint}}. \end{aligned} \tag{8}$$

Here, the KL divergence enforces policy regularization to prevent suboptimality, while the stepwise transitions encapsulate the rollouts within the learned latent world model $\mathcal{F}$ over a planning horizon H. The formulation in equation 8 anchors the policy optimization on the distribution β, thereby mitigating the failure modes characterized in Problems 1 and 2. Because the exact density of β is intractable, we introduce the following implicit energy function.

Learning Base Policy via Implicit Energy Function. To this end, the implicit energy function E_θ is learned from the buffer dataset via a contrastive objective:

$$\min_{E_\theta} \mathcal{L}(E_\theta) = -\mathbb{E}\Big[\log \frac{\exp(-E_\theta(z,a))}{\exp(-E_\theta(z,a)) + \sum_j \exp(-E_\theta(z, a_j^-))} \Big]. \tag{9}$$

As the dataset is large-scale and diverse, the negative samples a_j^- randomly drawn from other latent states within the same batch can be closely approximated by a uniform distribution over the entire action space. Consequently, the summation in the denominator asymptotically tracks the unweighted partition function, thereby aligning the learned energy function with the log behavior policy density up to a state-dependent constant $C(z)$, i.e., $-E(z,a) \approx \log \beta(a|z) + C(z)$. Based on the learned energy, we can reparameterize the reward function by absorbing the implicit energy function as $\tilde{\mathcal{R}}(z_h, a_h) = \mathcal{R}(z_h, a_h) - \frac{\eta}{\gamma^{h-t}} E(z_h, a_h)$, where η is the regularized factor, and replace $\mathcal{R}$ in equation 7 with $\tilde{\mathcal{R}}$.

Diffusion Policy. Instead of relying on MPPI merely for policy search, we adopt MBDPO to unify search and policy optimization. Unlike the conventional model-free setting, the score function ϕ in equation 3 can be estimated directly from trajectories imagined from the learned world model, without requiring a forward diffusion process:

$$\phi(z_t, a_{t:t+H}^\tau, \tau) = -\frac{a_{t:t+H}^\tau}{1-\bar{\alpha}^\tau} + \frac{\sqrt{\bar{\alpha}^\tau}}{1-\bar{\alpha}^\tau} \mathbb{E}_{a_{t:t+H}^{0|\tau} \sim \mathcal{N}(\frac{1}{\sqrt{\bar{\alpha}^\tau}} a_{t:t+H}^\tau, \frac{1-\bar{\alpha}^\tau}{\bar{\alpha}^\tau} I)} \big[w(a_{t:t+H}^{0|\tau}) \cdot a_{t:t+H}^{0|\tau} \big]. \tag{10}$$

See Lemma F.1 in Appendix F.3 for the detailed derivation of equations 10–12. Here, $a_{t:t+H}^{0|\tau}$ is a random variable that follows the sample posterior distribution $\mathcal{N}(\frac{1}{\sqrt{\bar{\alpha}^\tau}} a_{t:t+H}^\tau, \frac{1-\bar{\alpha}^\tau}{\bar{\alpha}^\tau} I)$, and $\bar{\alpha}^\tau = \prod_{k=1}^\tau \alpha^k$. The importance weight $w(a_{t:t+H}^{0|\tau})$ is defined as:

$$w(a_{t:t+H}^{0|\tau}) = \frac{\exp(\frac{\tilde{G}(z_t, a_{t:t+H}^{0|\tau})}{\kappa})}{\mathbb{E}_{a_{t:t+H}^{0|\tau} \sim \mathcal{N}(\frac{1}{\sqrt{\bar{\alpha}^\tau}} a_{t:t+H}^\tau, \frac{1-\bar{\alpha}^\tau}{\bar{\alpha}^\tau} I)} [\exp(\frac{\tilde{G}(z_t, a_{t:t+H}^{0|\tau})}{\kappa})]}, \tag{11}$$

where κ is the temperature factor, with

$$\tilde{G}(z_t, a_{t:t+H}^{0|\tau}) := \sum_{h=t}^{t+H-1} \gamma^{h-t} \tilde{\mathcal{R}}(z_h, a_h^{0|\tau}) + \gamma^H \hat{Q}(z_{t+H}^{0|\tau}, a_{t+H}^{0|\tau}) - \eta E(z_{t+H}^{0|\tau}, a_{t+H}^{0|\tau})$$

representing the cumulative return regularized with the implicit energy. By substituting the score function into the reverse-time transition kernel in equation 3, the action sequence $a^{\tau-1}_{t:t+H}$ at $\tau-1$ is

$$a^{\tau-1}_{t:t+H} = \frac{1}{\sqrt{\alpha^\tau}}\left(a^{\tau}_{t:t+H} + (1-\alpha^\tau)\phi(z_t, a^{\tau}_{t:t+H}, \tau)\right) + \sigma(\tau)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)\,. \tag{12}$$

To generate the optimized action sequence, we start from a Gaussian prior $a^{N}_{t:t+H} \sim \mathcal{N}(0, I)$ and iteratively apply the reverse-time transition kernel defined in equation 2 and equation 3. At each denoising step, the score in equation 10 shapes the transition kernel based on the returns evaluated from imagined trajectories within the world model. In this sense, the world model acts as a score-field corrector for policy optimization. Once the exact score function is matched in equation 10, the optimal policy for the KL-constrained objective in equation 8 is derived, as stated in the following theorem.

Theorem 1 (Equivalence of Score Matching and Policy Optimization). Consider the policy refinement task in equation 2 and the energy minimization objective in equation 9. For a given temperature $\kappa > 0$ and any latent state $z_t \in \mathcal{Z}$, let π^*_ϕ be the solution to the KL-constrained policy optimization problem formulated in equation 8. The optimum is attained if and only if the score matching condition in equation 10 is satisfied. Specifically, under the transition dynamics of the world model, the optimal policy π^*_ϕ can be approximated by the following Gibbs distribution:

$$\pi^*_\phi(a_{t:t+H}|z_t) \propto \left(\prod_{h=t}^{t+H} \beta(a_h|z_h)\right) \exp\left(\frac{G(z_t, a_{t:t+H})}{\kappa}\right) \prod_{h=t}^{t+H-1} \mathbb{1}_{\{z_{h+1}=\mathcal{F}(z_h,a_h)\}}, \tag{13}$$

where $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function enforcing the latent dynamics, and the partition function is omitted for brevity. A detailed proof is provided in Appendix F.4.

Theorem 1 provides a unified perspective that bridges search and policy optimization: rather than explicitly parameterizing a policy network, optimization can be implicitly executed through iterative rollouts and score-based sampling within the world model. The diffusion policy intrinsically bridges score matching within the world model and true policy optimization: the world model continuously bootstraps trajectories by generating imagined rollouts to facilitate score matching. Crucially, as established in Theorem 1, this score matching process is mathematically equivalent to performing true policy optimization. The optimal policy follows the Gibbs distribution in equation 13. MBDPO directly addresses the bottlenecks identified in Problems 1 and 2. We next describe how MBDPO implements this policy optimization in practice.

3.3 Practical Algorithm

For the practical implementation of MBDPO, we adopt a joint-training paradigm that synchronizes world model learning with diffusion policy optimization (see Algorithm 1 in Appendix H). We maintain a replay buffer $\mathcal{B}$ to store trajectory segments, supporting various training configurations including online, offline, and offline-to-online settings. The framework is jointly trained through an alternating optimization scheme that iteratively updates (i) the world model and value iteration, and (ii) the diffusion policy optimization.

(i) World Model and Value Iteration: Utilizing trajectory segments sampled from the buffer $\mathcal{B}$, we optimize the core components of the world model defined in equation 1, namely the *encoder* $\mathcal{E}$, *latent dynamics* $\mathcal{F}$, and *reward predictor* $\mathcal{R}$. Simultaneously, the *value function* $\hat{Q}$ is updated via one-step temporal difference (TD) learning, while the *implicit energy* E_θ is refined according to the objective in equation 9. The joint minimizer for $\mathcal{E}, \mathcal{F}, \mathcal{R}, \hat{Q}$ and E_θ can be written as

$$\min \mathbb{E}_{\tau\sim\mathcal{B}}[\sum_{h=t}^{t+H} \gamma^{h-t}(\|\mathcal{F}(z_h, a_h) - \text{sg}(\mathcal{E}(s_{h+1}))\|_2^2 + \text{CE}(\hat{r}_h, r(s_h, a_h)) + \text{TD}(\hat{Q}(z_h, a_h)) + \mathcal{L}(E_\theta))], \tag{14}$$

where $\text{sg}(\cdot)$ denotes the `stop-grad` operator and $\text{CE}(\cdot,\cdot)$ represents the cross-entropy loss, utilized to capture sparse or discrete rewards across diverse environments. Following Lillicrap et al. (2016), the one-step TD loss is defined as $\text{TD}(\hat{Q}(z_h, a_h)) \coloneqq \text{CE}\big(\hat{Q}(z_h, a_h), r_h + \gamma\bar{Q}(z_{h+1}, a_{h+1})\big)$, where $\bar{Q}$ is the target network maintained as an exponential moving average (EMA) of $\hat{Q}$. The energy loss $\mathcal{L}(E_\theta)$ is optimized according to equation 9, using actions sampled from other latent states as

negative pairs. Other training configurations, such as normalization and encoder architectures, strictly follow the TD-MPC framework to highlight the significance of scalable policy learning within world models.

(ii) Diffusion Policy Optimization. As shown in equation 10 and Theorem 1, the policy optimization relies on the imagined samples from the learned world models under a weighted expectation (the red-shaded term in equation 15). Since the expectation cannot be computed analytically, we approximate it using Monte Carlo sampling. MBDPO employs a weighted Monte Carlo estimator based on a proposal distribution $q(a^{0|\tau}_{t:t+H}|z_t)$, chosen as a Gaussian $\mathcal{N}(\frac{1}{\sqrt{\bar{\alpha}^\tau}}a^\tau_{t:t+H}, \frac{1-\bar{\alpha}^\tau}{\bar{\alpha}^\tau}I)$ according to equation 11. The asymptotically unbiased expectation can thus be approximated using T Monte Carlo samples $\{a^{0|\tau,(i)}_{t:t+H}\}^T_{i=1}$ drawn from $q(a^{0|\tau}_{t:t+H}|z_t)$:

$$\phi(z_t, a^\tau_{t:t+H}, \tau) \approx -\frac{a^\tau_{t:t+H}}{1-\bar{\alpha}^\tau} + \frac{\sqrt{\bar{\alpha}^\tau}}{1-\bar{\alpha}^\tau} \cdot \sum_{i=1}^T \; w(a^{0|\tau,(i)}_{t:t+H}) \cdot a^{0|\tau,(i)}_{t:t+H} \tag{15}$$

with $w(a^{0|\tau,(i)}_{t:t+H}) = \exp(\frac{\tilde{G}(z_t, a^{0|\tau,(i)}_{t:t+H})}{\kappa}) / \sum_{i=1}^T \exp(\frac{\tilde{G}(z_t, a^{0|\tau,(i)}_{t:t+H})}{\kappa})$ for samples $a^{0|\tau,(i)}_{t:t+H} \sim q(a^{0|\tau}_{t:t+H}|z_t)$. These weights rely on imagined trajectories rolled out by the learned latent dynamics $\mathcal{F}$ to evaluate the cumulative return $\tilde{G}$, as shown in Figure 1. To avoid the heavy computational overhead of online rollouts during deployment, this Monte Carlo estimation is exclusively employed during score network training, such that

$$\min \mathbb{E}\big[\|\hat{\phi}(z_t, a^\tau_{t:t+H}, \tau) - \phi(z_t, a^\tau_{t:t+H}, \tau)\|^2_2\big], \tag{16}$$

where $\hat{\phi}(z_t, a^\tau_{t:t+H}, \tau)$ is the learned score function. During inference, the score network serves as an amortized solution for efficient policy generation, plugging $\hat{\phi}$ into equation 12 for iterative sampling of the target action sequence. This framework benefits from provable theoretical guarantees: as established in (Chen et al., 2022; Cheng et al., 2026), the error relative to the target Gibbs distribution in equation 13 decays at a polynomial rate Poly(N, T, L_ϕ), where N, T, and L_ϕ denote the diffusion steps, Monte Carlo sample size, and the score function's Lipschitz constant, respectively.

4 Experiments

Benchmarks. We evaluate MBDPO in two complementary evaluation settings. (1) The TD-MPC2 protocol (Hansen et al., 2024) covers 121 diverse control tasks from four major domains, including DMControl (Tassa et al., 2018), MetaWorld (Yu et al., 2020), ManiSkill2 (Gu et al., 2023), and MyoSuite (Caggiano et al., 2022). We use this protocol to evaluate per-task online control, multi-task offline scaling, and offline-to-online adaptation. (2) MMBench, introduced by Newt (Hansen et al., 2026), is a large-scale benchmark for massively multi-task online reinforcement learning. It contains 200 tasks across ten domains, including DMControl, DMControl Extended, Meta-World, ManiSkill3 (Tao et al., 2025), MuJoCo (Todorov et al., 2012), MiniArcade (Hansen et al., 2026), Box2D (Brockman et al., 2016), RoboDesk (Kannan et al., 2021), OGBench (Park et al., 2025a), and Atari (Bellemare et al., 2013). A single agent is jointly trained across all tasks through online interaction. For a controlled comparison, MBDPO follows the Newt protocol and uses the same demonstrations, frozen task-language embeddings, and online interaction budget. We use this benchmark to evaluate massively multi-task online performance and scaling behavior.

Baselines. Under the TD-MPC2 protocol, we compare MBDPO against several state-of-the-art and data-efficient RL algorithms: (1) Soft Actor-Critic (SAC) (Haarnoja et al., 2018), a representative off-policy baseline; (2) DreamerV3 (Hafner et al., 2023), a leading world-model-based approach; (3) TD-MPC (Hansen et al., 2022), the first foundational work of the TD-MPC series; and (4) TD-MPC2 (Hansen et al., 2024), which shares its algorithmic core with recent advances in large-scale model-based RL (Hansen et al., 2026). On MMBench, we compare MBDPO with five representative baselines. These include (1) Newt (Hansen et al., 2026), a language-conditioned multitask world model with demonstration-based pretraining; (2) TD-MPC2 (Hansen et al., 2024), a multitask world-model baseline without language or demonstration supervision; (3) FastTD3 (Seo et al., 2025), an off-policy baseline for massively parallel RL; (4) PPO (Schulman et al., 2017), a standard on-policy baseline; and (5) single-task behavior cloning (BC) (Pomerleau, 1988; Atkeson & Schaal, 1997), trained separately for each task using demonstrations.

Result Analysis. Our analysis first examines whether MBDPO empirically mitigates the search–value misalignment. We then evaluate the algorithm's generalization across offline pretraining, online learning, and offline-to-online settings, as well as its scaling behavior. We also conduct a comprehensive evaluation of the computational efficiency of MBDPO.

(1) Does MBDPO reduce search–value misalignment?

Figure 2 compares cross-TD error and policy drift during training. Cross-TD error measures the discrepancy between the search-based TD targets and the targets used for value learning. MBDPO maintains lower and more stable cross-TD error than TD-MPC2, consistent with the Value Iteration Gap in Problem 1. Policy drift measures the average log-likelihood ratio between π and β. MBDPO keeps this drift closer to zero, indicating that policy improvement remains closer to the data-supported distribution. Together, these results show that MBDPO reduces the empirical signatures of search–value misalignment. Additional task-level diagnostics are provided in Appendix G.2.

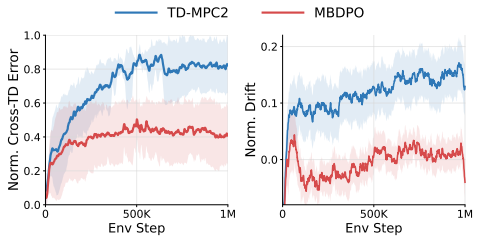

Figure 2: **Cross-TD error and policy drift.** Task-normalized averages across eight online tasks. Shaded regions indicate uncertainty.

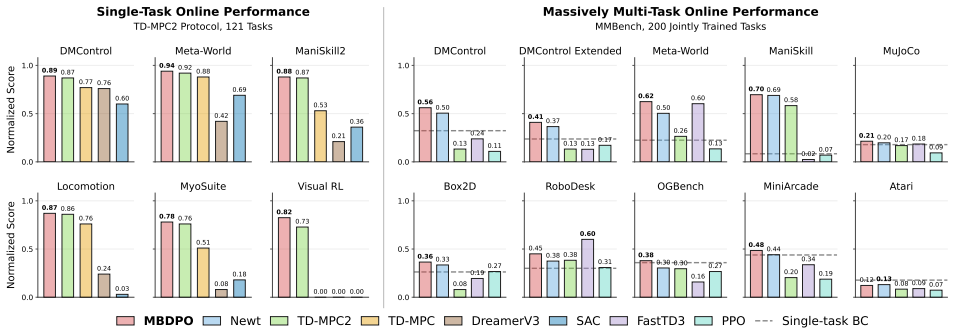

Figure 3: **Online control performance.** Normalized scores under the TD-MPC2 protocol (left) and MMBench (right). Dashed lines denote single-task BC.

(2) Does MBDPO perform strongly across both single-task and massively multi-task settings? Figure 3 summarizes performance under the two evaluation settings. Under the TD-MPC2 protocol, MBDPO achieves the highest aggregate score across all six evaluation groups, covering 121 independently trained tasks from four domains. On MMBench, a single MBDPO agent is jointly trained on 200 tasks and achieves the highest score in eight of the ten domains. These results demonstrate strong performance under both independent single-task training and massively multi-task online learning. Complete per-task learning curves are provided in Appendices G.2.2 and G.3.3.

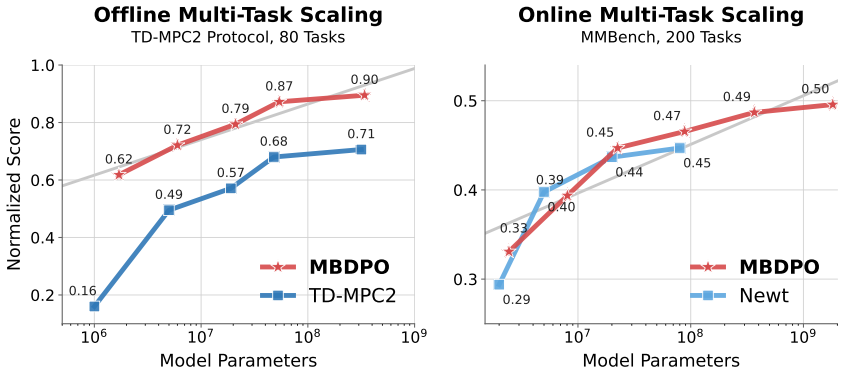

Figure 4: **Offline and online scaling.** Normalized scores versus model size under the 80-task offline protocol (left) and MMBench (right). Gray lines show least-squares log-linear fits to MBDPO.

(3) Does MBDPO scale across offline and online multi-task regimes? Figure 4 shows the relationship between model capacity and performance under the two benchmarks. In the 80-task offline setting, the

score of MBDPO increases from 0.62 to 0.90 as its capacity grows from 1.7M to 340M parameters, outperforming TD-MPC2 throughout the evaluated range. On MMBench, its score increases from 0.33 to 0.50 as the model scales from 2.4M to 1.82B parameters. MBDPO remains competitive with or outperforms Newt at comparable model sizes while continuing to benefit from larger capacities. The fitted trends provide complementary evidence of favorable scaling in both offline and online multi-task learning. Complete online scaling curves are provided in Appendix G.3.

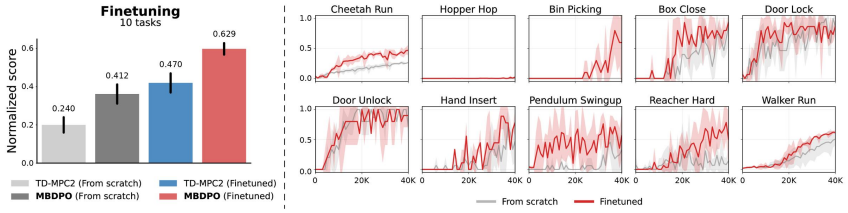

Figure 5: **Offline-to-online fine-tuning.** Aggregate performance over ten unseen tasks (left) and per-task learning curves with 40K online interactions (right). Models are pretrained on 70 tasks before fine-tuning.

(4) Can offline pretraining support efficient online adaptation? MBDPO naturally extends to offline-to-online adaptation. We pretrain the model on 70 tasks and fine-tune it on 10 unseen tasks using only 40K online interactions per task. As shown in Figure 5, fine-tuned MBDPO achieves an aggregate normalized score of 0.629, compared with 0.412 when trained from scratch and 0.470 for fine-tuned TD-MPC2. These results show that offline pretraining provides a useful initialization for subsequent online adaptation.

(5) Which design choices make MBDPO effective? Figure 6 examines behavior anchoring and sampling fidelity. Performance remains stable for moderate values of the energy weight η, whereas an excessively large value places too much emphasis on the behavior distribution and limits policy improvement. The strong result at $\eta = 0$ further indicates that diffusion policy optimization contributes beyond behavior anchoring alone. Increasing the Monte Carlo samples T and diffusion denoising steps N generally improves performance through more accurate score estimation and finer policy refinement, respectively. Increasing T affects training-time score estimation, whereas increasing N introduces an inference-time cost. These trends are consistent with the theoretical dependence on T and N. Extended results are provided in Appendix G.2.5.

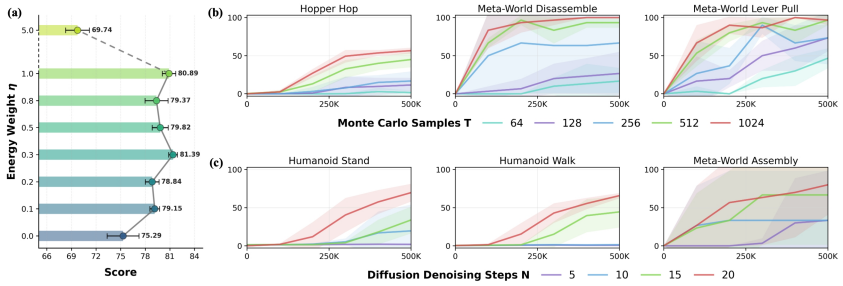

Figure 6: **Ablations.** (a) effects of the energy weight η, (b) Monte Carlo samples T, and (c) diffusion denoising steps N. Shading denotes variation across three seeds.

5 Conclusion

In this work, we identified *search–value misalignment* as a structural bottleneck in world-model reinforcement learning, arising when policy search and value learning operate under different policy distributions. To address this problem, we introduced Model-Based Diffusion Policy Optimization (MBDPO), which reformulates trajectory search as behavior-anchored diffusion policy optimization within the learned world model. MBDPO reduces the empirical signatures of this misalignment, achieves strong performance across single-task and massively multi-task benchmarks, exhibits favorable scaling in both offline and online regimes, and supports efficient offline-to-online adaptation. More broadly, these results suggest that scaling world-model RL may require treating search not as an external planner, but as part of policy optimization itself.

AI USE STATEMENT

Generative AI tools were used to improve the readability and clarity of parts of the manuscript and to assist with editing software code. Generative AI tools were not used to formulate mathematical claims, develop theoretical arguments, design the research methodology, interpret experimental results, or generate synthetic datasets. All AI-assisted text and code were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

REFERENCES

Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 12–20. Morgan Kaufmann, 1997.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013. doi: 10.1613/jair.3912.

Richard Bellman. *Stability theory of differential equations*. Courier Corporation, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. Myosuite–a contact-rich simulation suite for musculoskeletal motor control. *arXiv preprint arXiv:2205.13600*, 2022.

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pp. 510–517. IEEE, 2015.

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

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. *arXiv preprint arXiv:2601.21306*, 2026.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

Xiaoyuan Cheng, Xiaohang Tang, and Yiming Yang. Safe and stable control via lyapunov-guided diffusion models. *arXiv preprint arXiv:2509.25375*, 2025.

Xiaoyuan Cheng, Wenxuan Yuan, Boyang Li, Yuanchao Xu, Yiming Yang, Hao Liang, Bei Peng, Robert Loftin, Zhuo Sun, and Yukun Hu. How does the lagrangian guide safe reinforcement learning through diffusion models? *arXiv preprint arXiv:2602.02924*, 2026.

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

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. *Advances in Neural Information Processing Systems*, 33:14156–14170, 2020.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. *arXiv preprint arXiv:2502.11612*, 2025.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Rémi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. *arXiv preprint arXiv:2503.09817*, 2025.

Jay W Forrester. Counterintuitive behavior of social systems. *Theory and decision*, 2(2):109–140, 1971.

Haotian Fu, Shangqun Yu, Michael Littman, and George Konidaris. Model-based lifelong reinforcement learning with bayesian exploration. *Advances in Neural Information Processing Systems*, 35: 32369–32382, 2022.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. *arXiv preprint arXiv:2501.16142*, 2025.

Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. Learning and leveraging world models in visual representation learning. *arXiv preprint arXiv:2403.00504*, 2024.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023.

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

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

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*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025a.

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

Jianhua Han, Meng Tian, Jiangtong Zhu, Fan He, Huixin Zhang, Sitong Guo, Dechang Zhu, Hao Tang, Pei Xu, Yuze Guo, et al. Percept-wam: Perception-enhanced world-awareness-action model for robust end-to-end autonomous driving. *arXiv preprint arXiv:2511.19221*, 2025.

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2024.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. *arXiv preprint arXiv:2511.19584*, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. In *International Conference on Learning Representations*, 2026.

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

Physical Intelligence, Ali Amin Bo, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: A Steerable Generalist Robotic Foundation Model with Emergent Capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Harini Kannan, Danijar Hafner, Chelsea Finn, and Dumitru Erhan. Robodesk: A multi-task reinforcement learning benchmark. https://github.com/google-research/robodesk, 2021.

Leonid Kuvayev and Richard S. Sutton. Model-based reinforcement learning with an approximate, learned model. In *Proceedings of the ninth Yale workshop on adaptive and learning systems*, volume 1996, pp. 101–105, 1996.

Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

Haotian Lin, Pengcheng Wang, Jeff Schneider, and Guanya Shi. Td-m(pc)^2: Improving temporal difference mpc through policy constraint, 2025. URL https://arxiv.org/abs/2502.03550.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

Michel Ma, Tianwei Ni, Clement Gehring, Pierluca D'Oro, and Pierre-Luc Bacon. Do transformer world models give better policy gradients? *arXiv preprint arXiv:2402.05290*, 2024.

Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1):159–195, 1996.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

Chaoyi Pan, Zeji Yi, Guanya Shi, and Guannan Qu. Model-based diffusion for trajectory optimization. *Advances in Neural Information Processing Systems*, 37:57914–57943, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, 2025a.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *arXiv preprint arXiv:2506.04168*, 2025b.

Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1, pp. 305–313, 1988.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

Michael Psenka, Michael Rabbat, Aditi Krishnapriyan, Yann LeCun, and Amir Bar. Parallel stochastic gradient-based planning for world models. *arXiv preprint arXiv:2602.00475*, 2026.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

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

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

Yutaka Shimizu and Masayoshi Tomizuka. Bisimulation metric for model predictive control. *arXiv preprint arXiv:2410.04553*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Jyothir SV, Siddhartha Jalagam, Yann LeCun, and Vlad Sobal. Gradient-based planning with world models. *arXiv preprint arXiv:2312.17227*, 2023.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

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

Sebastian Thrun, Knut Möller, and Alexander Linden. Planning with an adaptive world model. *Advances in neural information processing systems*, 3, 1990.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Hado P Van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. *arXiv preprint arXiv:2503.18871*, 2025.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017.

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

Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4974–4981. IEEE, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

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

Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. *arXiv preprint arXiv:2511.00423*, 2025.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Appendix Overview

The appendix complements the main paper with additional background, detailed theoretical analysis, complete experimental results, and implementation details. Its organization is summarized below.

- **Appendix B. Notation.** We summarize the notation used throughout the paper and appendix.
- **Appendix C. More Literature Review.** We discuss additional related work on world models and diffusion policies.
- **Appendix D. Limitations and Future Directions.** We discuss the current limitations of MBDPO and several directions for future work.
- **Appendix E. Technical Definitions.** We provide additional background on diffusion policies and formally define the Bellman operator used in our analysis.
- **Appendix F. Theoretical Analysis.** We present the complete theoretical development of MBDPO, including the policy-search mechanism used by the TD-MPC series, detailed analyses of the Value Iteration Gap and Policy Improvement Gap, the derivation of the exact score function, and the proof of Theorem 1.
- **Appendix G. Additional Experimental Results.** We provide complete results under two evaluation settings.
  - **Appendix G.2. Experiments under the TD-MPC2 Protocol.** We report the training environments, offline multi-task scaling, single-task online performance, search–value misalignment diagnostics, offline-to-online adaptation, additional ablations, representation analyses, and computational cost.
  - **Appendix G.3. Experiments on MMBench.** We report online multi-task scaling, domain-level comparisons, and complete learning curves for all 200 tasks across ten domains.
- **Appendix H. Practical Implementation and Hyperparameters.** We provide the complete MBDPO algorithm, describe the online, offline, and offline-to-online training regimes, and report the architecture and hyperparameter settings used in the experiments.

Together, these appendices provide the background, theoretical foundations, empirical evidence, and implementation details that complement the main paper.

B NOTATION

Notation	Meaning
a	action
e	learnable task embedding
s	state
t	time step
r	reward function
z	latent state
$\mathcal{A}$	space of action
E	implicit energy function
$\mathcal{E}$	encoder of world model
$\mathcal{E}_{env}$	space of learnable task embedding
$\mathcal{F}$	latent dynamics of world model
G	cumulative return over an H-step action rollout
$\tilde{G}$	energy-regularized cumulative return over an H-step action rollout
J	standard discounted returns
$\mathcal{M}$	Markov decision process
$\mathcal{S}$	space of state
$\mathcal{T}$	transition dynamics
Q	Q value function
$\hat{Q}$	value function learned from buffer dataset
$\mathcal{R}$	latent reward function
$\tilde{\mathcal{R}}$	energy-regularized latent reward function
$T^{\square}$	Bellman update
$\mathcal{Z}$	space of latent state
$\alpha, \bar{\alpha}$	scalar factor of noise injection
β	behavioral policy distribution
κ	temperature
π	policy
η	energy weight
ρ	initial distribution
$\sigma^2(\tau)$	variance induced by the reverse diffusion transition kernel
π_ϕ	policy generated by score function
ϕ	denoted symbol of score function
γ	discount factor
$\mathbb{1}_{\{\cdot\}}$	indicator function

C More Literature Review

World Models. The concept of world models originates from Jay W. Forrester, who introduced it for constructing mental representations of complex systems (Forrester, 1971). It was later adapted to machine learning, enabling agents to learn and act within learned internal environments (Ha & Schmidhuber, 2018). Later, the Dreamer series (Hafner et al., 2019; 2020; 2025a;b) transformed latent world modeling from a proof-of-concept into a scalable and robust RL paradigm. However, due to model bias, training the policy purely within the learned world model can lead to distributional shift (Janner et al., 2019). Another line of research, TD-MPC (Hansen et al., 2022; 2024; 2026), combines model-based and model-free RL. However, the existence of misalignment between search and value hinders scalability and performance. Recent approaches, including Vision-Language-Action models with world models (Zhen et al., 2024; Cen et al., 2025; Team et al., 2026; Intelligence et al., 2026) and World Action Models (WAMs) (Han et al., 2025; Ye et al., 2026; Li et al., 2026), have been proposed to generate action chunks and state representations, driven by the rapid development of large pre-trained vision models. Compared to RL, the WAM family does not perform explicit policy optimization, which limits its ability to achieve long-horizon control and robust generalization.

Diffusion Policy. Diffusion policies can be broadly categorized into model-free and model-based approaches. Model-free diffusion policies do not rely on learned dynamics; instead, they directly learn the policy distribution from a fixed dataset (Janner et al., 2022; Chi et al., 2025) or a learned Q-function (Psenka et al., 2023; Dong et al., 2025; Cheng et al., 2026; Li & Levine, 2026). In contrast, model-based diffusion techniques leverage environment dynamics to estimate the score function without relying on offline data (Pan et al., 2024; Xue et al., 2025; Cheng et al., 2025). Crucially, computing this score function requires access to the transition dynamics, making world models an ideal substrate; they provide the simulated environment necessary. However, in many practical scenarios, ground-truth dynamics are unavailable, limiting their direct applicability. This bottleneck, in turn, highlights the critical necessity of learned world models. By approximating a diverse spectrum of dynamics, world models hold the key to unlocking the full potential of model-based diffusion for scalable policy learning. Despite this immense promise, effective model-based diffusion policies within learned world models remain underexplored.

D Limitations and Future Directions

While MBDPO demonstrates strong scaling properties and performance, several avenues remain for future exploration. First, due to bounded computational resources, we have not yet scaled our framework to extremely large parameter sizes or massive foundation-model scales; investigating its full scaling limits remains a direct next step. Second, our evaluations are currently focused on simulated environments. Moving forward, deploying MBDPO on real-world robots will be critical to testing its robust transferability and practical viability. Finally, an exciting prospective direction is to integrate MBDPO with even more powerful, pretrained representations (e.g., visual or multimodal foundation models as encoders). Leveraging such richer visual priors could improve the latent representation and unlock truly large-scale RL within world models.

E TECHNICAL DEFINITIONS

In this section, we provide further technical details and formal definitions to supplement the descriptions in the main text.

E.1 DIFFUSION POLICIES BASED ON DDPM

We briefly review the fundamental mechanism of Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020; Song et al., 2020). In our setting, the diffusion process based on DDPM is used for sampling and generating target policies. Let $\tau \in \{1, \dots, N\}$ denote the diffusion timestep and $\bar{\alpha}^\tau \in (0, 1]$ be the cumulative noise schedule. The forward and reverse diffusion processes for the action sequence are defined as:

$$\text{(Forward)} \quad p(a^\tau_{t:t+H} | a^0_{t:t+H}, z_t) = \mathcal{N}\left(a^\tau_{t:t+H}; \sqrt{\bar{\alpha}^\tau} a^0_{t:t+H}, (1 - \bar{\alpha}^\tau) I\right), \tag{17}$$

$$\text{(Reverse)} \quad a^{\tau-1}_{t:t+H} = \frac{1}{\sqrt{\alpha^\tau}} \left(a^\tau_{t:t+H} + (1 - \alpha^\tau)\phi(z_t, a^\tau_{t:t+H}, \tau)\right) + \sigma(\tau)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{18}$$

where $p(a^\tau_{t:t+H} | a^0_{t:t+H}, z_t)$ is the transition kernel in the forward process, and $\phi(z_t, a^\tau_{t:t+H}, \tau) = \nabla_{a^\tau_{t:t+H}} \log \pi^\tau(a^\tau_{t:t+H} | z_t)$ denotes the score function at diffusion step τ. In our framework, as policy optimization is performed within the latent world model, the explicit forward diffusion process is not strictly required for training. Nevertheless, the forward process remains essential for computing the transition kernel in the reverse process, as formulated in equation 3.

E.2 BELLMAN OPERATOR

To formally analyze value estimation within the world model, we define the Bellman operator as follows (Sutton et al., 1998):

Definition E.1 (Bellman Operator)**.** Consider the space of Q-functions $\mathcal{Q}$ equipped with the L_∞ norm[3]. For a given policy π, the Bellman evaluation operator $T^\pi : \mathcal{Q} \to \mathcal{Q}$ is defined as:

$$(T^\pi Q)(z_t, a_t) = \mathcal{R}(z_t, a_t) + \gamma \mathbb{E}_{z_{t+1} \sim \mathcal{F}(z_t, a_t), a_{t+1} \sim \pi}[Q(z_{t+1}, a_{t+1})], \tag{19}$$

where $\mathcal{R}(z_t, a_t)$ denotes the latent reward function, $\mathcal{F}$ represents the latent dynamics within the learned world model, and $\gamma \in (0, 1)$ is the discount factor.

This operator serves exclusively for policy evaluation and value function update. By repeatedly applying T^π, the Q-function converges to Q^π (according to the Banach fixed-point theorem (Bellman, 2013)), the true value of the current policy π. This evaluation stage is crucial as it provides an accurate value estimate, which MBDPO then uses to guide the policy optimization process toward higher-reward regions.

[3]Formally, $\mathcal{Q} = \{Q | Q : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}\}$ is the set of all bounded real-valued functions defined on the latent state-action space. The L_∞ norm is defined as $\|Q\|_\infty = \sup_{(z,a) \in \mathcal{Z} \times \mathcal{A}} |Q(z, a)|$.

F THEORETICAL ANALYSIS

In this section, we provide a theoretical analysis of the proposed framework. We first examine the misalignment between search and value learning, highlighting how discrepancies between the search policy and the learned value function can affect optimization. We then analyze MBDPO from a theoretical perspective, characterizing its learning dynamics and discussing why it can mitigate such misalignment. Finally, we present an error analysis of the overall algorithm, identifying the key sources of approximation error and their impact on the resulting policy.

F.1 BOTTLENECKS IN THE EXISTING PARADIGM

Before presenting our main results, we first provide a detailed mathematical analysis of policy search in the TD-MPC series (Hansen et al., 2022; 2025; Zhan et al., 2025; Lin et al., 2025).

F.1.1 HOW DOES POLICY SEARCH WORK IN THE TD-MPC SERIES?

The policy search procedure can be decomposed into two steps:

- **Step 1: Initialize with a Prior Policy.** The search distribution is initialized with an isotropic Gaussian prior policy $p_0(\cdot|z) \sim \mathcal{N}(\mu(z), \sigma^2)$ (usually parameterized by a policy network), where the mean μ is estimated by maximizing the objective in equation 6.
- **Step 2: Policy Search via Model Predictive Path Integral Planning (MPPI).** Starting from the initialized prior policy $\mathcal{N}(\mu(z), \sigma^2)$, the algorithm iteratively samples candidate action sequences and evaluates them using model-based rollouts (Williams et al., 2017). These samples are then reweighted according to their estimated returns (abstracted in equation 20),

$$\underbrace{p_0(a_{t:t+H}|z_t)}_{\text{Gaussian prior}} \xrightarrow{\text{sample}} \underbrace{\{a^{(i)}_{t:t+H}\}^T_{i=1}}_{\text{candidate actions}} \xrightarrow{\text{evaluated by } H\text{-step } \hat{Q}} \underbrace{\{\hat{Q}^{(i)}\}^T_{i=1}}_{\text{estimated values}} \xrightarrow{\text{reweight}} \underbrace{\pi(a_{t:t+H}|z_t)}_{\text{search target}}, \tag{20}$$

  where T represents the sample number, which yields an improved policy target through importance-weighted aggregation.

This procedure highlights a key limitation of policy search in the TD-MPC series. The resulting target policy is induced by the search distribution rather than obtained through an explicit policy optimization problem constrained around an anchored policy distribution. As a result, the searched target can become effectively uncontrolled: its update is primarily determined by the return estimates used for importance weighting. When the learned value function or model-based return estimates are inaccurate, erroneously high Q-values may assign excessive weights to suboptimal action sequences. Consequently, the aggregated policy target can be biased toward poor actions, causing policy learning to imitate a bad search-induced target rather than ensuring a monotonic improvement over the prior policy. In this sense, the search step does not constitute a principled policy optimization procedure; it constructs a heuristic target from weighted samples of learned world models, without explicitly optimizing a regularized policy objective or enforcing an improvement guarantee.

F.1.2 ANALYSIS OF BOTTLENECK 1

We begin with a theoretical analysis of the Bellman update. In the setting of a latent world model, the $\hat{Q}$ value function is learned from the (behavioral) policy β (parameterized with a policy network).

Proof. By Definition E.1, for any latent state-action pair (z_t, a_t), we have

$$(T^\pi \hat{Q})(z_t, a_t) = \mathcal{R}(z_t, a_t) + \gamma \mathbb{E}_{z_{t+1} \sim \mathcal{F}(z_t, a_t)} \mathbb{E}_{a_{t+1} \sim \pi(\cdot|z_{t+1})} [\hat{Q}(z_{t+1}, a_{t+1})], \tag{21}$$

and

$$(T^\beta \hat{Q})(z_t, a_t) = \mathcal{R}(z_t, a_t) + \gamma \mathbb{E}_{z_{t+1} \sim \mathcal{F}(z_t, a_t)} \mathbb{E}_{a_{t+1} \sim \beta(\cdot|z_{t+1})} [\hat{Q}(z_{t+1}, a_{t+1})]. \tag{22}$$

Therefore,

$$\begin{aligned}
&\left|(T^{\pi}\hat{Q})(z_t,a_t)-(T^{\beta}\hat{Q})(z_t,a_t)\right| \\
&=\gamma\left|\mathbb{E}_{z_{t+1}\sim\mathcal{F}(z_t,a_t)}\left[\sum_{a_{t+1}}\left(\pi(a_{t+1}|z_{t+1})-\beta(a_{t+1}|z_{t+1})\right)\hat{Q}(z_{t+1},a_{t+1})\right]\right| \\
&\leq\gamma\mathbb{E}_{z_{t+1}\sim\mathcal{F}(z_t,a_t)}\left[\left|\sum_{a_{t+1}}\left(\pi(a_{t+1}|z_{t+1})-\beta(a_{t+1}|z_{t+1})\right)\hat{Q}(z_{t+1},a_{t+1})\right|\right] \\
&\leq\gamma\|\hat{Q}\|_{\infty}\mathbb{E}_{z_{t+1}\sim\mathcal{F}(z_t,a_t)}\left[\int_{a_{t+1}}\left|\pi(a_{t+1}|z_{t+1})-\beta(a_{t+1}|z_{t+1})\right|da_{t+1}\right] \\
&=\gamma\|\hat{Q}\|_{\infty}\mathbb{E}_{z_{t+1}\sim\mathcal{F}(z_t,a_t)}\left[\|\pi(\cdot|z_{t+1})-\beta(\cdot|z_{t+1})\|_1\right]. \qquad (23)
\end{aligned}$$

By Pinsker's inequality,

$$\|\pi(\cdot|z_{t+1})-\beta(\cdot|z_{t+1})\|_1\leq\sqrt{2D_{\mathrm{KL}}\left(\pi(\cdot|z_{t+1})\|\beta(\cdot|z_{t+1})\right)}. \qquad (24)$$

Thus,

$$\left|(T^{\pi}\hat{Q})(z_t,a_t)-(T^{\beta}\hat{Q})(z_t,a_t)\right|\leq\gamma\|\hat{Q}\|_{\infty}\mathbb{E}_{z_{t+1}\sim\mathcal{F}(z_t,a_t)}\left[\sqrt{2D_{\mathrm{KL}}\left(\pi(\cdot|z_{t+1})\|\beta(\cdot|z_{t+1})\right)}\right]. \qquad (25)$$

Taking the supremum over all latent state-action pairs $(z_t,a_t)\in\mathcal{Z}\times\mathcal{A}$ gives

$$\begin{aligned}
\|T^{\pi}\hat{Q}-T^{\beta}\hat{Q}\|_{\infty}&\leq\gamma\|\hat{Q}\|_{\infty}\sup_{z_t,a_t}\mathbb{E}_{z_{t+1}\sim\mathcal{F}(z_t,a_t)}\left[\sqrt{2D_{\mathrm{KL}}\left(\pi(\cdot|z_{t+1})\|\beta(\cdot|z_{t+1})\right)}\right] \\
&\leq\gamma\|\hat{Q}\|_{\infty}\sqrt{2D_{\mathrm{KL}}^{\max}(\pi\|\beta)}, \qquad (26)
\end{aligned}$$

with $D_{\mathrm{KL}}^{\max}(\pi\|\beta):=\sup_{z\in\mathcal{Z}}D_{\mathrm{KL}}(\pi(\cdot|z)\|\beta(\cdot|z))$, which completes the proof. □

F.2 Analysis of Bottleneck 2

Proof. We consider the latent MDP induced by the world model transition $z_{t+1}=\mathcal{F}(z_t,a_t)$. Equivalently, the transition kernel is deterministic:

$$P_{\mathcal{F}}(z'|z,a)=\mathbb{1}_{\{z'=\mathcal{F}(z,a)\}}, \qquad (27)$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. Let d^{π} and d^{β} be the discounted latent-state occupancy distributions under policies π and β, respectively.

By the performance difference lemma (Mahadevan, 1996), the true improvement of π over β can be written as

$$J(\pi)-J(\beta)=\frac{1}{1-\gamma}\mathbb{E}_{z\sim d^{\pi},a\sim\pi(\cdot|z)}\left[A^{\beta}(z,a)\right], \qquad (28)$$

where

$$A^{\beta}(z,a)=Q^{\beta}(z,a)-V^{\beta}(z),\qquad V^{\beta}(z)=\mathbb{E}_{a\sim\beta(\cdot|z)}[Q^{\beta}(z,a)]. \qquad (29)$$

Equivalently,

$$J(\pi)-J(\beta)=\frac{1}{1-\gamma}\mathbb{E}_{z\sim d^{\pi}}\left[\mathbb{E}_{a\sim\pi(\cdot|z)}Q^{\beta}(z,a)-\mathbb{E}_{a\sim\beta(\cdot|z)}Q^{\beta}(z,a)\right]. \qquad (30)$$

We now decompose the right-hand side into a surrogate term evaluated under d^{β} and a distribution-shift term:

$$J(\pi)-J(\beta)=\frac{1}{1-\gamma}\mathbb{E}_{z\sim d^{\beta}}\left[\mathbb{E}_{a\sim\pi(\cdot|z)}Q^{\beta}(z,a)-\mathbb{E}_{a\sim\beta(\cdot|z)}Q^{\beta}(z,a)\right]+\Delta_{\mathrm{occ}}, \qquad (31)$$

where

$$\Delta_{\mathrm{occ}}=\frac{1}{1-\gamma}\left(\mathbb{E}_{z\sim d^{\pi}}-\mathbb{E}_{z\sim d^{\beta}}\right)\left[\mathbb{E}_{a\sim\pi(\cdot|z)}Q^{\beta}(z,a)-\mathbb{E}_{a\sim\beta(\cdot|z)}Q^{\beta}(z,a)\right]. \qquad (32)$$

Under the standard boundedness assumption $\|Q^\beta\|_\infty \le Q_{\max}$, the distribution-shift term can be bounded by a trust-region style argument (Schulman et al., 2015):

$$|\Delta_{\text{occ}}| \le C_1 D_{\text{KL}}^{\max}(\pi\|\beta), \tag{33}$$

where

$$D_{\text{KL}}^{\max}(\pi\|\beta) := \sup_z D_{\text{KL}}\left(\pi(\cdot|z)\|\beta(\cdot|z)\right). \tag{34}$$

This bound follows from the fact that the discrepancy between the occupancy distributions d^π and d^β is controlled by the maximum policy divergence at each latent state, together with Pinsker's inequality.

Therefore,

$$J(\pi) - J(\beta) \ge \frac{1}{1-\gamma}\mathbb{E}_{z\sim d^\beta}\left[\mathbb{E}_{a\sim\pi(\cdot|z)}Q^\beta(z,a) - \mathbb{E}_{a\sim\beta(\cdot|z)}Q^\beta(z,a)\right] - C_1 D_{\text{KL}}^{\max}(\pi\|\beta). \tag{35}$$

Next, we relate the true value function Q^β to its approximation $\hat{Q}$. For any policy μ, we have

$$\left|\mathbb{E}_{z\sim d^\beta, a\sim\mu(\cdot|z)}\left[Q^\beta(z,a) - \hat{Q}(z,a)\right]\right| \le \|Q^\beta - \hat{Q}\|_\infty. \tag{36}$$

Applying this inequality to both $\mu = \pi$ and $\mu = \beta$ gives

$$\begin{aligned}&\mathbb{E}_{z\sim d^\beta}\left[\mathbb{E}_{a\sim\pi(\cdot|z)}Q^\beta(z,a) - \mathbb{E}_{a\sim\beta(\cdot|z)}Q^\beta(z,a)\right]\\ &\ge \mathbb{E}_{z\sim d^\beta}\left[\mathbb{E}_{a\sim\pi(\cdot|z)}\hat{Q}(z,a) - \mathbb{E}_{a\sim\beta(\cdot|z)}\hat{Q}(z,a)\right] - 2\|Q^\beta - \hat{Q}\|_\infty.\end{aligned} \tag{37}$$

By the definition of the surrogate objective, the first term on the right-hand side is exactly

$$\hat{J}_\beta(\pi) - \hat{J}_\beta(\beta). \tag{38}$$

Thus,

$$J(\pi) - J(\beta) \ge \frac{1}{1-\gamma}\left(\hat{J}_\beta(\pi) - \hat{J}_\beta(\beta)\right) - C_1 D_{\text{KL}}^{\max}(\pi\|\beta) - C_2\|Q^\beta - \hat{Q}\|_\infty, \tag{39}$$

where $C_1, C_2 = \frac{2}{1-\gamma} > 0$ are constants depending on γ and the boundedness of the value function. This completes the proof. □

Implication: search policy should be anchored to the non-search behavioral policy. The above two results reveal a common source of failure when the search policy π is optimized without being anchored to the behavioral policy β. From equation 4, the Bellman update induced by π deviates from the Bellman update induced by β by an amount controlled by the policy divergence:

$$\|T^\pi\hat{Q} - T^\beta\hat{Q}\|_\infty \le \gamma\|\hat{Q}\|_\infty\sqrt{2D_{\text{KL}}^{\max}(\pi\|\beta)}. \tag{40}$$

This means that when π moves far away from β, the value function $\hat{Q}$, which is learned from data generated by β, is no longer updated consistently with the distribution on which it was trained. As a result, applying T^π can propagate value estimates into latent regions that are poorly covered by the behavioral policy, where $\hat{Q}$ may be inaccurate.

Meanwhile, equation 5 shows that the true policy improvement is lower bounded by the estimated improvement only up to two error terms:

$$J(\pi) - J(\beta) \ge \frac{1}{1-\gamma}\left(\hat{J}_\beta(\pi) - \hat{J}_\beta(\beta)\right) - C_1 D_{\text{KL}}^{\max}(\pi\|\beta) - C_2\|Q^\beta - \hat{Q}\|_\infty. \tag{41}$$

The first penalty is the TRPO-style distribution-shift term (Schulman et al., 2015), which indicates that policy improvement is inherently local around the behavioral (or reference) policy β. The second penalty captures the value estimation gap introduced by using the learned value function $\hat{Q}$ instead of the true value function Q^β. Together, these two terms imply that a large surrogate improvement under $\hat{J}_\beta(\pi)$ does not necessarily translate into true improvement when π deviates substantially from β.

Therefore, the two bottlenecks reinforce each other. An unconstrained search policy may exploit spurious high-value regions of $\hat{Q}$ and move away from the data-supported behavior distribution. This increases the KL divergence between π and β, enlarges the Bellman update mismatch in equation 4, and further increases the value estimation error appearing in equation 5. Consequently, the search procedure can enter a self-reinforcing cycle of distribution shift and value overestimation, where inaccurate value estimates attract the policy, and the shifted policy further amplifies value errors.

This analysis suggests that search in latent world models should not be performed as an unconstrained policy improvement step. Instead, the search policy should be regularized or anchored to the behavioral policy β, for example through a KL constraint or a policy-prior term. Such an anchoring mechanism keeps the search policy within the region where the learned value function is reliable, thereby reducing Bellman mismatch, controlling distribution shift, and enabling more stable monotonic improvement within the world model.

F.3 Derivation of Exact Score Function in equation 10

Lemma F.1 (Exact Score Function)**.** *Consider the forward and reverse diffusion processes defined in equation 17 and equation 18 in the latent world model over $\mathcal{Z}$. For a latent state z_t and a noisy action sequence $a^\tau_{t:t+H}$ at diffusion step τ, define a proposal distribution*

$$q^\tau(a^0_{t:t+H}|a^\tau_{t:t+H}, z_t) = \mathcal{N}\left(a^0_{t:t+H}; \frac{1}{\sqrt{\bar\alpha^\tau}} a^\tau_{t:t+H}, \frac{1-\bar\alpha^\tau}{\bar\alpha^\tau} I\right). \tag{42}$$

Then the exact score function with respect to $a^\tau_{t:t+H}$ can be written as

$$\phi(z_t, a^\tau_{t:t+H}, \tau) = -\frac{a^\tau_{t:t+H}}{1-\bar\alpha^\tau} + \frac{\sqrt{\bar\alpha^\tau}}{1-\bar\alpha^\tau} \mathbb{E}_{a^0_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H})}\left[w(a^{0|\tau}_{t:t+H}) a^{0|\tau}_{t:t+H}\right], \tag{43}$$

where the normalized importance weight is

$$w(a^{0|\tau}_{t:t+H}) = \frac{\exp\left(\tilde G(z_t, a^0_{t:t+H})/\kappa\right)}{\mathbb{E}_{\tilde a^0_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H})}\left[\exp\left(\tilde G(z_t, \tilde a^0_{t:t+H})/\kappa\right)\right]}. \tag{44}$$

Proof. We start from the definition of the score function. For a fixed latent state z_t, the marginal distribution of the noisy action sequence $a^\tau_{t:t+H}$ is given by

$$\pi^\tau_\phi(a^\tau_{t:t+H}|z_t) = \int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}. \tag{45}$$

Therefore,

$$\begin{aligned}
\phi(z_t, a^\tau_{t:t+H}, \tau) &= \nabla_{a^\tau_{t:t+H}} \log \pi^\tau_\phi(a^\tau_{t:t+H}|z_t) \\
&= \frac{\nabla_{a^\tau_{t:t+H}} \int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}}{\int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}} \\
&= \frac{\int \nabla_{a^\tau_{t:t+H}} p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}}{\int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}}.
\end{aligned} \tag{46}$$

Under the DDPM forward process, the transition kernel is

$$\begin{aligned}
p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t) &= \mathcal{N}\left(a^\tau_{t:t+H}; \sqrt{\bar\alpha^\tau} a^0_{t:t+H}, (1-\bar\alpha^\tau) I\right) \\
&\propto \exp\left(-\frac{1}{2} \frac{\left(a^\tau_{t:t+H} - \sqrt{\bar\alpha^\tau} a^0_{t:t+H}\right)^\top \left(a^\tau_{t:t+H} - \sqrt{\bar\alpha^\tau} a^0_{t:t+H}\right)}{1-\bar\alpha^\tau}\right).
\end{aligned} \tag{47}$$

Hence,

$$\nabla_{a^\tau_{t:t+H}} p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t) = -\frac{a^\tau_{t:t+H} - \sqrt{\bar\alpha^\tau} a^0_{t:t+H}}{1-\bar\alpha^\tau} p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t). \tag{48}$$

Substituting this identity into equation 46, we obtain

$$\begin{aligned}
\phi(z_t, a^\tau_{t:t+H}, \tau) &= \frac{\int -\frac{a^\tau_{t:t+H} - \sqrt{\bar\alpha^\tau} a^0_{t:t+H}}{1-\bar\alpha^\tau} p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}}{\int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}} \\
&= -\frac{a^\tau_{t:t+H}}{1-\bar\alpha^\tau} + \frac{\sqrt{\bar\alpha^\tau}}{1-\bar\alpha^\tau} \frac{\int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) a^0_{t:t+H} da^0_{t:t+H}}{\int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}}.
\end{aligned} \tag{49}$$

Thus, computing the exact score reduces to estimating the posterior mean

$$\mathbb{E}\left[a^0_{t:t+H}|a^\tau_{t:t+H}, z_t\right] = \frac{\int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) a^0_{t:t+H} da^0_{t:t+H}}{\int p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)\pi^0_\phi(a^0_{t:t+H}|z_t) da^0_{t:t+H}}. \tag{50}$$

To express this posterior mean using proposal sampling, we introduce the Gaussian proposal distribution

$$q_\tau(a^0_{t:t+H}|a^\tau_{t:t+H}, z_t) = \mathcal{N}\left(a^0_{t:t+H}; \frac{1}{\sqrt{\bar{\alpha}^\tau}} a^\tau_{t:t+H}, \frac{1-\bar{\alpha}^\tau}{\bar{\alpha}^\tau} I\right). \tag{51}$$

This proposal is proportional to the DDPM likelihood $p(a^\tau_{t:t+H}|a^0_{t:t+H}, z_t)$ when viewed as a function of $a^0_{t:t+H}$. Therefore, equation 50 can be rewritten as

$$\mathbb{E}\left[a^0_{t:t+H}|a^\tau_{t:t+H}, z_t\right] = \frac{\mathbb{E}_{a^{0|\tau}_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H}, z_t)}\left[\pi^0_\phi(a^{0|\tau}_{t:t+H}|z_t) a^{0|\tau}_{t:t+H}\right]}{\mathbb{E}_{a^{0|\tau}_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H}, z_t)}\left[\pi^0_\phi(a^{0|\tau}_{t:t+H}|z_t)\right]}. \tag{52}$$

Substituting the posterior mean back into the score expression gives

$$\phi(z_t, a^\tau_{t:t+H}, \tau) = -\frac{a^\tau_{t:t+H}}{1-\bar{\alpha}^\tau} + \frac{\sqrt{\bar{\alpha}^\tau}}{1-\bar{\alpha}^\tau} \mathbb{E}_{a^0_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H}, z_t)}\left[w(a^{0|\tau}_{t:t+H}) a^{0|\tau}_{t:t+H}\right], \tag{53}$$

where the normalized importance weight is defined as

$$w(a^{0|\tau}_{t:t+H}) = \frac{\pi^0_\phi(a^{0|\tau}_{t:t+H}|z_t)}{\mathbb{E}_{\tilde{a}^{0|\tau}_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H}, z_t)}\left[\pi^0_\phi(\tilde{a}^{0|\tau}_{t:t+H}|z_t)\right]}. \tag{54}$$

Finally, focusing on the closed-form optimal policy to the regularized objective in Eq. 8:

$$\pi^0_\phi(a^{0|\tau}_{t:t+H}|z_t) \propto \exp\left(\frac{\tilde{G}(z_t, a^{0|\tau}_{t:t+H})}{\kappa}\right). \tag{55}$$

Hence, the normalized importance weight becomes

$$w(a^{0|\tau}_{t:t+H}) = \frac{\exp\left(\tilde{G}(z_t, a^{0|\tau}_{t:t+H})/\kappa\right)}{\mathbb{E}_{\tilde{a}^{0|\tau}_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H}, z_t)}\left[\exp\left(\tilde{G}(z_t, \tilde{a}^{0|\tau}_{t:t+H})/\kappa\right)\right]}. \tag{56}$$

This completes the proof. □

What is the role of the world model in the computation of the score function? The world model provides the latent dynamics used to evaluate each candidate clean action sequence sampled during the score computation. In the derivation above, the proposal distribution $q_\tau(a^{0|\tau}_{t:t+H} \mid a^\tau_{t:t+H}, z_t)$ is obtained from the DDPM forward kernel and is used to sample clean action sequences $a^{0|\tau}_{t:t+H}$ that are compatible with the noisy action sequence $a^\tau_{t:t+H}$. For each sampled action sequence, the world model recursively rolls out the latent dynamics

$$z^{0|\tau}_{h+1} = \mathcal{F}(z^{0|\tau}_h, a^{0|\tau}_h), \qquad h = t, \ldots, t+H-1, \tag{57}$$

starting from the current latent state $z^{0|\tau}_t = z_t$. This rollout produces a latent trajectory

$$\left(z_t, a^{0|\tau}_t, z^{0|\tau}_{t+1}, a^{0|\tau}_{t+1}, \ldots, z^{0|\tau}_{t+H}, a^{0|\tau}_{t+H}\right), \tag{58}$$

which is then used to compute the trajectory-level return

$$\tilde{G}(z_t, a^{0|\tau}_{t:t+H}) \coloneqq \sum_{h=t}^{t+H-1} \gamma^{h-t} \tilde{\mathcal{R}}(z^{0|\tau}_h, a^{0|\tau}_h) + \gamma^H \hat{Q}(z^{0|\tau}_{t+H}, a^{0|\tau}_{t+H}) - \eta E(z^{0|\tau}_{t+H}, a^{0|\tau}_{t+H}). \tag{59}$$

The return $\tilde{G}(z_t, a^{0|\tau}_{t:t+H})$ determines the normalized importance weight

$$w(a^{0|\tau}_{t:t+H}) = \frac{\exp\left(\tilde{G}(z_t, a^{0|\tau}_{t:t+H})/\kappa\right)}{\mathbb{E}_{\tilde{a}^{0|\tau}_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H}, z_t)}\left[\exp\left(\tilde{G}(z_t, \tilde{a}^{0|\tau}_{t:t+H})/\kappa\right)\right]}. \tag{60}$$

Thus, the world model does not directly define the Gaussian proposal used for denoising. Instead, it provides the model-based evaluation signal that reweights the sampled clean action sequences according to their predicted trajectory returns. The exact score is then computed as a return-weighted posterior mean:

$$\phi(z_t, a^\tau_{t:t+H}, \tau) = -\frac{a^\tau_{t:t+H}}{1-\bar{\alpha}^\tau} + \frac{\sqrt{\bar{\alpha}^\tau}}{1-\bar{\alpha}^\tau}\mathbb{E}_{a^{0|\tau}_{t:t+H}\sim q_\tau(\cdot|a^\tau_{t:t+H}, z_t)}\left[w(a^{0|\tau}_{t:t+H})a^{0|\tau}_{t:t+H}\right]. \tag{61}$$

In this sense, the world model converts each sampled clean action sequence into a trajectory-level value estimate, and the score function shifts the noisy action sequence toward clean action sequences that achieve higher predicted return under the latent dynamics.

A more straightforward abstraction is that the reverse transition kernel defines an iterative policy refinement process. Starting from a simple Gaussian prior over action sequences, the reverse diffusion process gradually transforms this prior into the optimized action-sequence policy:

$$\underbrace{\pi^N_\phi(a^N_{t:t+H}|z_t)}_{\text{Gaussian prior}} \xrightarrow{\pi^{N-1}_\phi(a^{N-1}_{t:t+H}|a^N_{t:t+H}, z_t)} \pi^{N-1}_\phi(a^{N-1}_{t:t+H}|z_t)\cdots \xrightarrow{\pi^0_\phi(a^0_{t:t+H}|a^1_{t:t+H}, z_t)} \underbrace{\pi^0_\phi(a^0_{t:t+H}|z_t)}_{\text{clean policy}}. \tag{62}$$

Equivalently, the final refined policy is obtained by marginalizing over all intermediate denoising steps:

$$\pi^0_\phi(a^0_{t:t+H}|z_t) = \int_{a^{1:N}_{t:t+H}} \pi^N_\phi(a^N_{t:t+H}|z_t)\prod_{\tau=N}^{1}\pi^{\tau-1}_\phi(a^{\tau-1}_{t:t+H}|a^\tau_{t:t+H}, z_t)da^{1:N}_{t:t+H}. \tag{63}$$

At each reverse step, the transition kernel $\pi^{\tau-1}_\phi(a^{\tau-1}_{t:t+H}|a^\tau_{t:t+H}, z_t)$ is guided by the score function $\phi(z_t, a^\tau_{t:t+H}, \tau)$. This score is computed by sampling candidate clean action sequences from the DDPM posterior proposal $q_\tau(\cdot|a^\tau_{t:t+H}, z_t)$, rolling them out through the world model, and reweighting them according to their predicted returns. *Consequently, the world model redefines the target for score matching by introducing model-based evaluations. This shift directly propagates to the stepwise transition kernel, thereby rendering the entire reverse diffusion process equivalent to policy optimization.*

From this perspective, reverse diffusion can be viewed as model-based policy optimization in action-sequence space. The Gaussian prior provides exploration over possible action sequences, while the world-model-evaluated score progressively reshapes this prior into a policy that concentrates on action sequences with higher predicted return. Therefore, the policy refinement in equation 2 is not merely a generative denoising procedure; it is also a KL-regularized policy optimization process where the world model supplies the trajectory-level objective used to guide each denoising step (stated in Theorem 1).

F.4 Proof of Theorem 1

Proof. We prove Theorem 1 in two steps: (1) demonstrating that the optimal distribution of the KL-constrained policy optimization problem formulated in equation 8 follows a Gibbs distribution, and (2) showing that score matching within the world model is equivalent to sampling from this optimal policy.

Step 1. Deriving the Optimal Gibbs Policy.

For a fixed latent state z_t, the latent trajectory is deterministically generated by the learned world model

$$z_{h+1} = \mathcal{F}(z_h, a_h), \qquad h = t, \ldots, t+H-1. \tag{64}$$

Therefore, once z_t and the action sequence $a_{t:t+H}$ are given, the trajectory return $G(z_t, a_{t:t+H})$ is completely determined by the rollout of the world model. Equivalently, the world-model constraint restricts feasible trajectories to those satisfying $z_{h+1} = \mathcal{F}(z_h, a_h)$.

We first consider the Lagrangian formulation of the KL-constrained policy optimization problem equation 8. Let $\eta > 0$ be the KKT multiplier associated with the KL divergence constraint. The regularized objective is:

$$\max_{\pi_\phi(\cdot|z_t)} \left\{\mathbb{E}_{a_{t:t+H}\sim\pi_\phi(\cdot|z_t)}\left[G(z_t, a_{t:t+H})\right] - \eta D_{\mathrm{KL}}\left(\pi_\phi(\cdot|z_t)\|\beta(\cdot|z_t)\right)\right\}. \tag{65}$$

Expanding the KL term, the objective becomes

$$\mathcal{J}(\pi_\phi) = \int \pi_\phi(a_{t:t+H}|z_t) G(z_t, a_{t:t+H}) da_{t:t+H} - \eta \int \pi_\phi(a_{t:t+H}|z_t) \log \frac{\pi_\phi(a_{t:t+H}|z_t)}{\beta(a_{t:t+H}|z_t)} da_{t:t+H}. \tag{66}$$

Note that the behavior prior β over the trajectory incorporates the deterministic latent dynamics constraint of world models:

$$\beta(a_{t:t+H}|z_t) = \prod_{h=t}^{t+H} \beta(a_h|z_h) \cdot \prod_{h=t}^{t+H-1} \mathbb{1}_{\{z_{h+1}=\mathcal{F}(z_h,a_h)\}}. \tag{67}$$

To enforce the probability normalization constraint $\int \pi_\phi(a_{t:t+H}|z_t) da_{t:t+H} = 1$, we introduce a second Lagrange multiplier λ and define the full Lagrangian:

$$\mathcal{L}(\pi_\phi, \lambda) = \int \pi_\phi(a_{t:t+H}|z_t) G(z_t, a_{t:t+H}) da_{t:t+H} - \eta \int \pi_\phi(a_{t:t+H}|z_t) \log \frac{\pi_\phi(a_{t:t+H}|z_t)}{\beta(a_{t:t+H}|z_t)} da_{t:t+H} + \lambda \left(\int \pi_\phi(a_{t:t+H}|z_t) da_{t:t+H} - 1 \right). \tag{68}$$

Taking the functional derivative with respect to $\pi_\phi(a_{t:t+H}|z_t)$ and setting it to zero gives

$$0 = G(z_t, a_{t:t+H}) - \eta \left(\log \frac{\pi_\phi(a_{t:t+H}|z_t)}{\beta(a_{t:t+H}|z_t)} + 1 \right) + \lambda. \tag{69}$$

Rearranging for the density ratio yields

$$\log \frac{\pi_\phi(a_{t:t+H}|z_t)}{\beta(a_{t:t+H}|z_t)} = \frac{G(z_t, a_{t:t+H})}{\eta} + \frac{\lambda - \eta}{\eta}. \tag{70}$$

Exponentiating both sides, we obtain the optimal policy:

$$\pi^*_\phi(a_{t:t+H}|z_t) = \frac{1}{Z(z_t)} \beta(a_{t:t+H}|z_t) \exp\left(\frac{G(z_t, a_{t:t+H})}{\eta} \right), \tag{71}$$

where $Z(z_t)$ is the partition function. Substituting the factorized form of β, the optimal distribution can be expressed as:

$$\pi^*_\phi(a_{t:t+H}|z_t) \propto \prod_{h=t}^{t+H} \beta(a_h|z_h) \cdot \exp\left(\frac{G(z_t, a_{t:t+H})}{\eta} \right) \cdot \prod_{h=t}^{t+H-1} \mathbb{1}_{\{z_{h+1}=\mathcal{F}(z_h,a_h)\}}. \tag{72}$$

Step 2. Generation of the Gibbs Distribution via Score Matching.

By Lemma F.1, an expression of the form equation 10 is analytically identical to the exact score function $\nabla_{a^\tau_{t:t+H}} \log \pi^\tau_\phi(a^\tau_{t:t+H}|z_t)$ of a forward-diffused distribution, whose underlying base distribution $\pi^0_\phi(a^0_{t:t+H}|z_t)$ is proportional to the unnormalized numerator of the weight $w(a^0_{t:t+H})$ in equation 44. Therefore, according to the properties of variance-preserving stochastic differential equations (Song et al., 2020), the score matching condition implicitly defines the following base distribution:

$$\pi^0_\phi(a^0_{t:t+H}|z_t) \propto \exp\left(\frac{\tilde{G}(z_t, a^0_{t:t+H})}{\eta} \right). \tag{73}$$

We now relate this implicit base distribution back to the policy derived in Step 1. Recall that the energy-regularized reward is defined as:

$$\tilde{\mathcal{R}}(z_h, a_h) = \mathcal{R}(z_h, a_h) - \frac{\eta}{\gamma^{h-t}} E(z_h, a_h). \tag{74}$$

Based on the energy minimization objective equation 9, the learned energy approximates the behavior prior up to a state-dependent constant: $-E(z_h, a_h) \approx \log \beta(a_h|z_h) + C(z_h)$. In continuous state spaces with continuous dynamics, the state variations over a short rollout horizon H are relatively small. Thus, we can reasonably approximate $C(z_h) \approx C(z_t)$, which acts as a fixed constant with respect to the future action sequence $a_{t:t+H}$. Summing over the trajectory and accounting for the discount factor, the energy-regularized return $\tilde{G}$ decomposes into the original return G and the behavior prior term:

$$\frac{\tilde{G}(z_t, a^0_{t:t+H})}{\eta} \approx \frac{G(z_t, a^0_{t:t+H})}{\eta} + \sum_{h=t}^{t+H} \log \beta(a_h|z_h) + (H+1)C(z_t). \tag{75}$$

Substituting this decomposition into equation 73, the constant term $(H+1)C(z_t)$ is absorbed into the proportionality constant, and the base distribution can be explicitly written as:

$$\pi^0_\phi(a^0_{t:t+H}|z_t) \propto \left(\prod_{h=t}^{t+H} \beta(a_h|z_h)\right) \exp\left(\frac{G(z_t, a^0_{t:t+H})}{\eta}\right) \prod_{h=t}^{t+H-1} \mathbb{1}_{\{z_{h+1}=\mathcal{F}(z_h, a^0_h)\}}. \tag{76}$$

This distribution is exactly identical to the optimal Gibbs policy $\pi^*_\phi(a_{t:t+H}|z_t)$ derived in Step 1.

Therefore, if the score matching condition is satisfied, the reverse diffusion process uses the exact score of the noisy marginal induced by the Gibbs policy. Consequently, the reverse diffusion sampler generates clean action sequences exactly from the optimal KL-regularized policy distribution.

Conversely, suppose the KL-constrained policy optimization problem attains its optimum. Then, by the variational argument above, the optimal policy has the Gibbs form in equation 71. The diffusion marginal induced by this clean-action distribution has a unique exact score, and by Lemma F.1, this score is precisely the expression in equation 10. Thus, the score matching condition is satisfied.

Combining both directions, the KL-constrained policy optimization problem attains its Gibbs-form optimum if and only if the score matching condition in equation 10 is satisfied. Finally, to provide flexibility in controlling the entropy and return trade-off during practical sampling, *we relax the strict Lagrange multiplier η to a general temperature parameter $\kappa > 0$*, yielding the target Gibbs distribution presented in Theorem 1. This completes the proof. □

G Additional Experimental Results

G.1 Training Environments

We evaluate MBDPO under two complementary settings. The TD-MPC2 protocol covers 121 tasks from four domains, including DMControl, MetaWorld, ManiSkill2, and MyoSuite (Tassa et al., 2018; Yu et al., 2020; Gu et al., 2023; Caggiano et al., 2022). It includes independently trained online agents, multi-task offline pretraining, and offline-to-online adaptation.

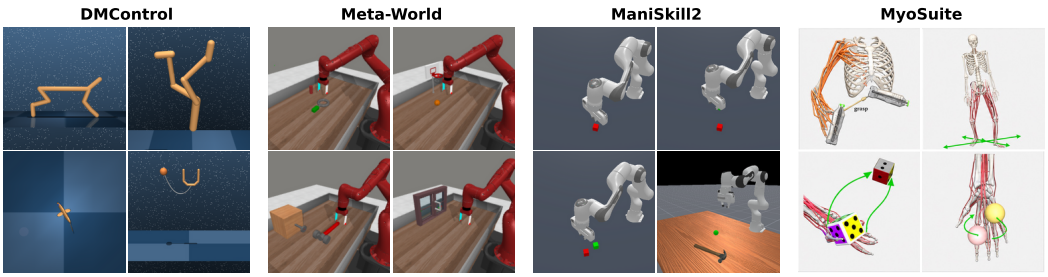

Figure 7: **Representative environments from the TD-MPC2 protocol.** Each domain is represented by four tasks.

MMBench (Hansen et al., 2026) evaluates massively multi-task online learning using a single agent jointly trained on 200 tasks across ten domains, including DMControl, DMControl Extended, Meta-World, ManiSkill, MuJoCo, Box2D, RoboDesk, OGBench, MiniArcade, and Atari.

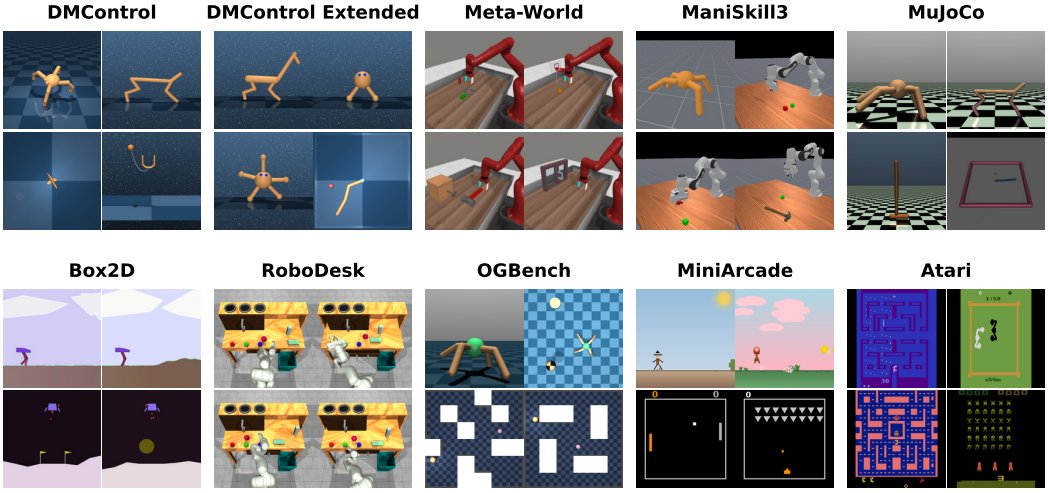

Figure 8: **Representative environments from MMBench.** Each domain is represented by four tasks.

The following subsections report detailed results under the two evaluation settings.

G.2 Experiments under the TD-MPC2 Protocol

G.2.1 Offline Multi-Task Scaling

Figure 9 provides the detailed offline scaling results for MBDPO and TD-MPC2 on the 30-task and 80-task datasets.

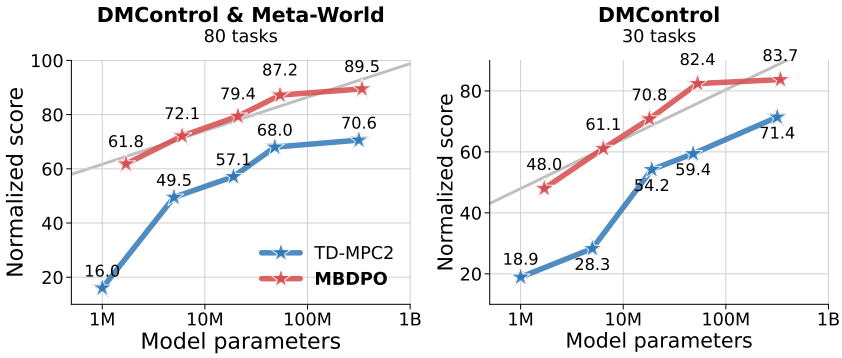

Figure 9: **Offline multi-task scaling.** Normalized scores on the 30-task and 80-task datasets across model sizes. MBDPO shows stronger overall scaling than TD-MPC2, with its largest gains at smaller model sizes.

G.2.2 Single-Task Online Performance

We evaluate MBDPO across 121 tasks from 4 major benchmarks (Figures 11-16), with each task averaged over 3 random seeds. Compared to both model-free and model-based baselines, our framework demonstrates superior final rewards, higher sample efficiency, and greater training stability of policy optimization.

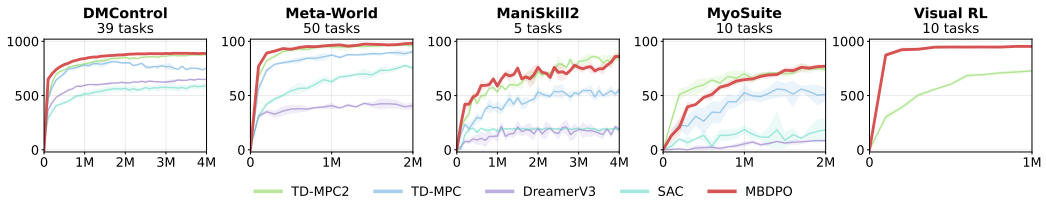

Figure 10: **Aggregate performance across four domains in the online setting.** DMControl, MetaWorld, ManiSkill2, and MyoSuite.

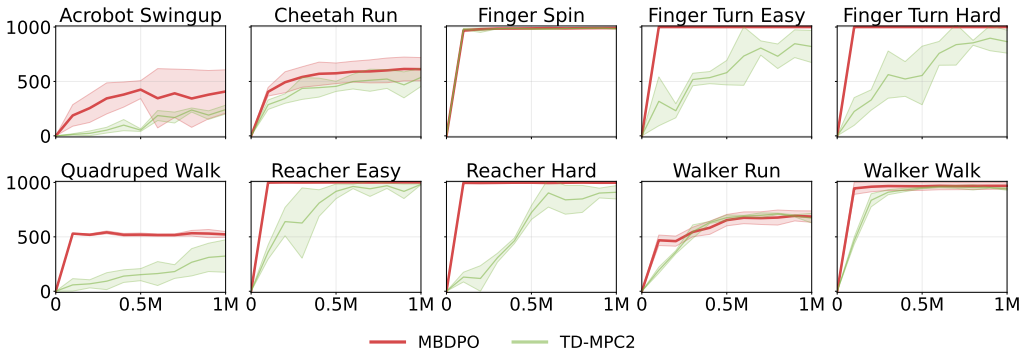

Figure 11: **Single-task Visual RL results.** Episode return as a function of environment steps. The first 1M environment steps are shown for each task.

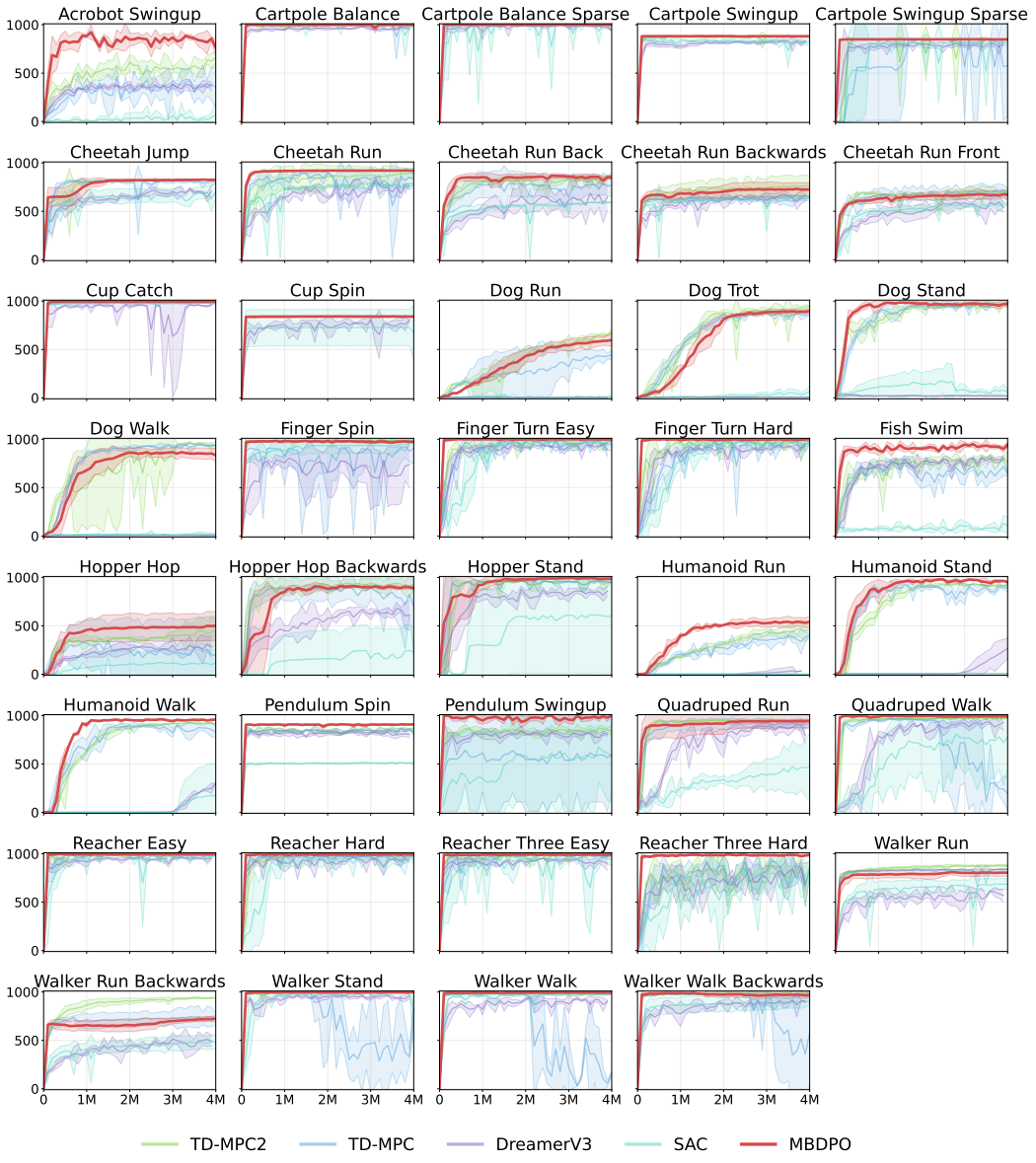

Figure 12: **Single-task DMControl results.** Episode return as a function of environment steps. The first 4M environment steps are shown for each task.

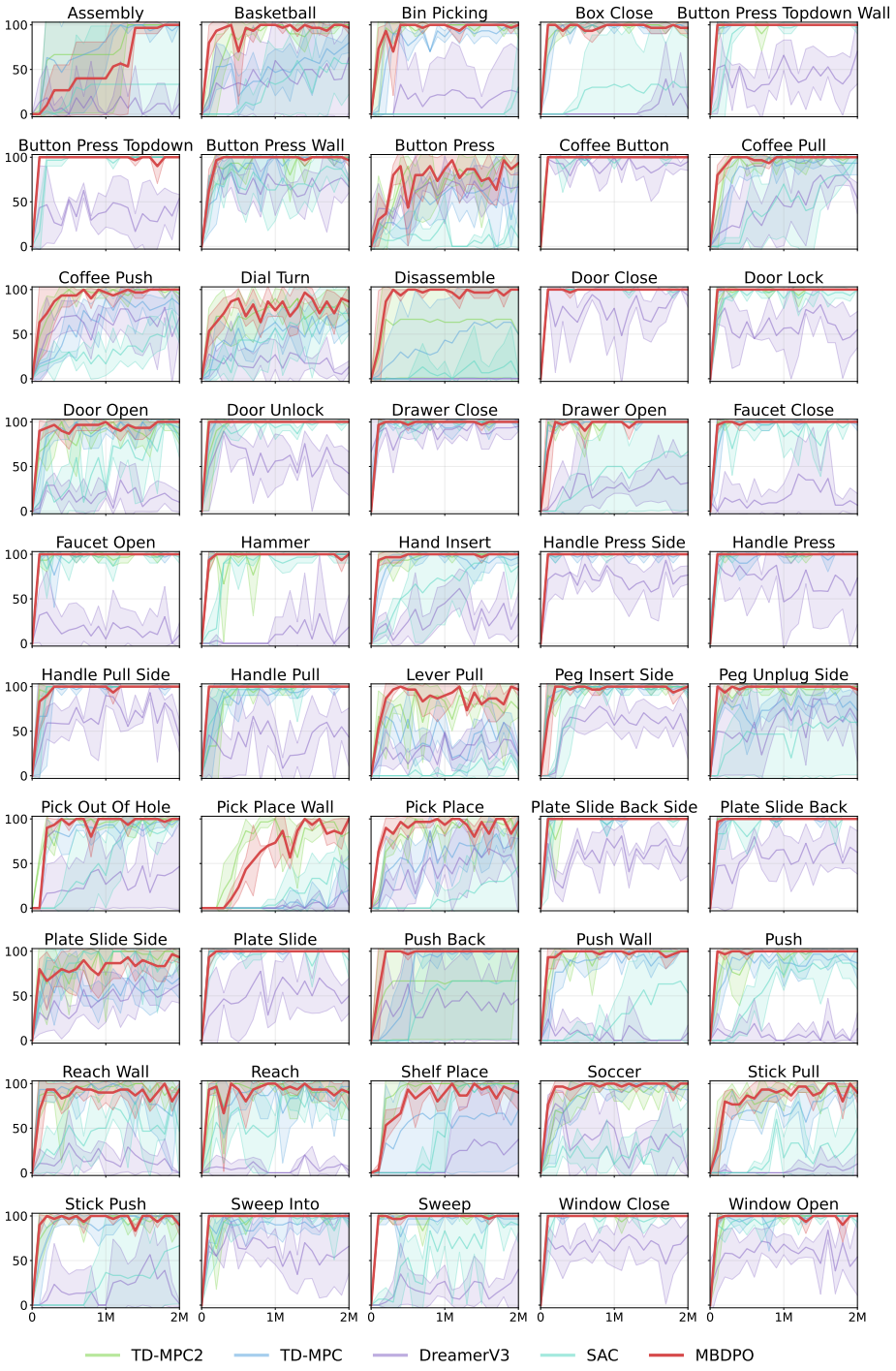

Figure 13: **Single-task MetaWorld results.** Success rate (%) as a function of environment steps. MBDPO achieves the best averaged performance over all MetaWorld tasks, while outperforming other methods on hard tasks such as *Pick Place Wall* and *Shelf Place*. DreamerV3 often fails to converge.

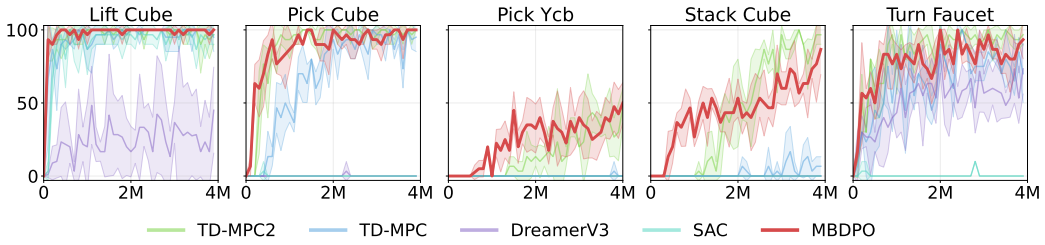

Figure 14: **Single-task ManiSkill2 results.** Success rate (%) as a function of environment steps on 5 object manipulation tasks from ManiSkill2. Pick YCB is the hardest task and involves manipulation of all 74 objects from the YCB (Calli et al., 2015). We report 4M environment steps for each task. MBDPO achieves a success rate above 75% on the Pick YCB task, whereas other methods fail to learn within the given budget.

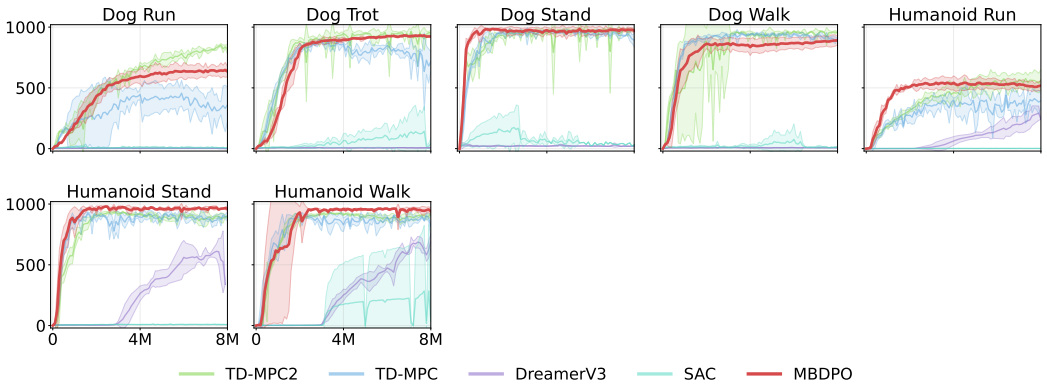

Figure 15: **High-dimensional locomotion results.** Episode return as a function of environment steps on all 7 "Locomotion" benchmark tasks.

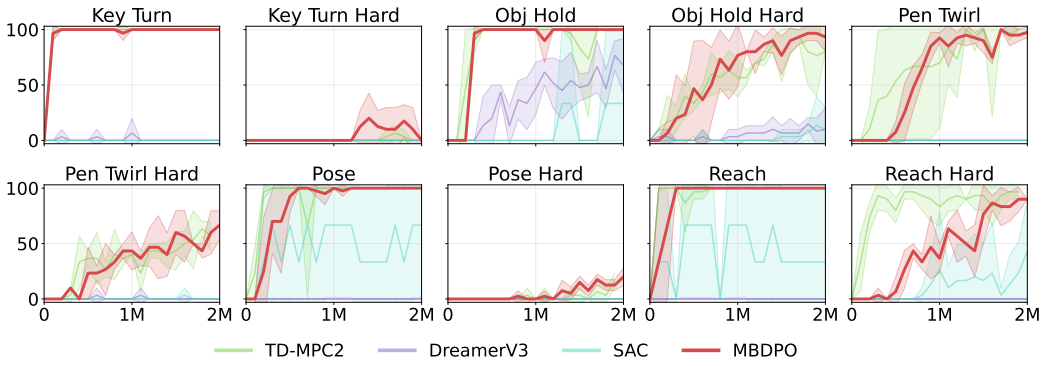

Figure 16: **Single-task MyoSuite results.** Success rate (%) as a function of environment steps. This task domain includes high-dimensional contact-rich musculoskeletal motor control with a physiologically accurate robot hand. Goals are randomized in tasks designated as "Hard". MBDPO achieves comparable or better performance than existing methods on all tasks from this benchmark.

G.2.3 SEARCH–VALUE MISALIGNMENT DIAGNOSTICS

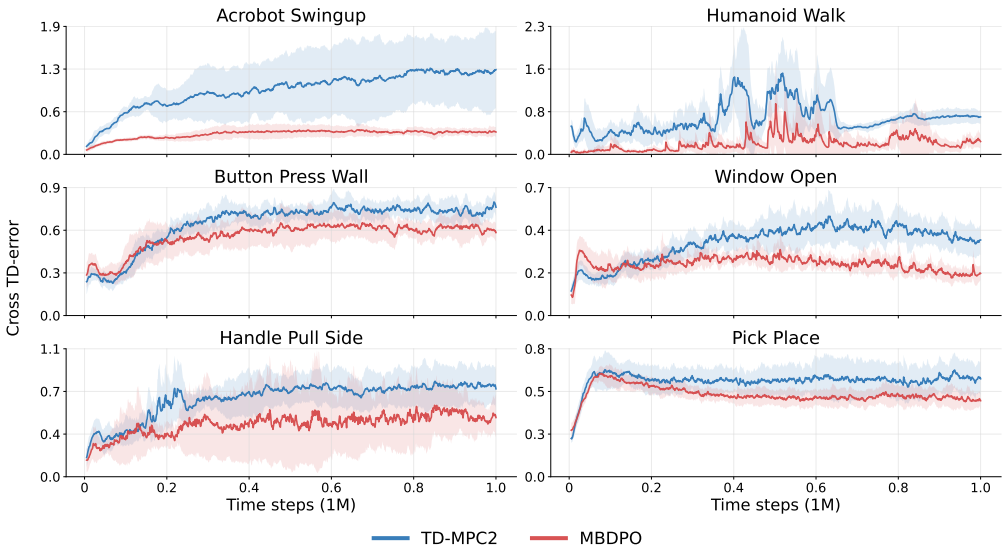

Figure 17: **Per-task cross TD-error during training.** EMA-smoothed cross TD-errors are reported for representative online tasks. The cross TD-error measures the temporal-difference error incurred when evaluating trajectories under a policy different from the one used for value-function training, thereby reflecting the distribution mismatch between policy improvement and value learning. Across most tasks, MBDPO exhibits lower cross TD-error than TD-MPC2, suggesting that diffusion policy optimization reduces out-of-distribution value queries and leads to more stable value iteration during training.

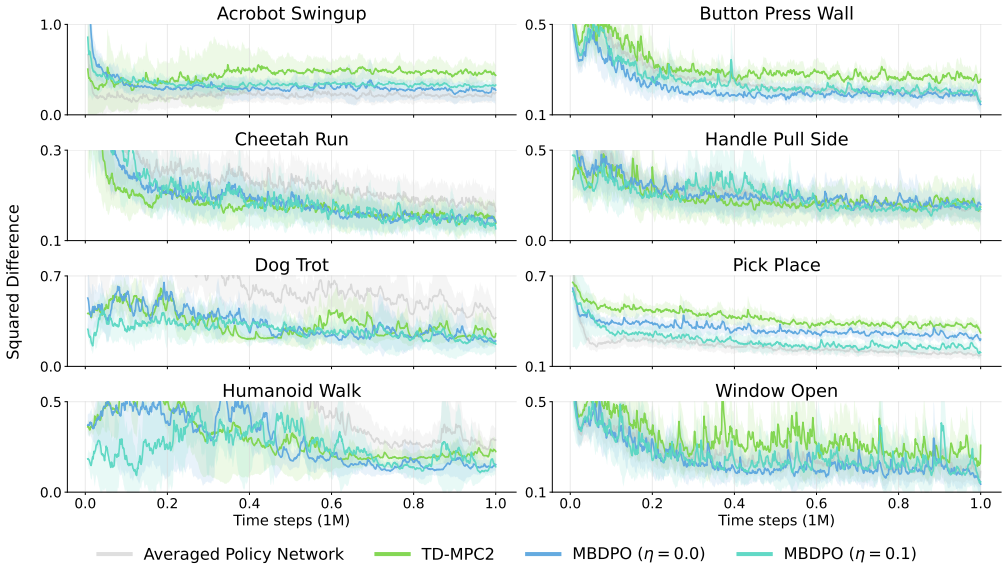

Figure 18: **Per-task action drift during training.** EMA-smoothed action differences are reported for each of the 8 online tasks. The Averaged Policy Network (Grey) curve reports the mean action drift of the policy networks from TD-MPC2 and the two MBDPO variants. Across these tasks, MBDPO exhibits smaller drift than TD-MPC2, indicating improved temporal stability under diffusion policy optimization.

G.2.4 OFFLINE-TO-ONLINE ADAPTATION

To evaluate transfer, we pre-train MBDPO on 70 tasks and fine-tune it on 10 unseen tasks using $40k$ online interactions per task. Figure 5 compares fine-tuning with training from scratch and with TD-MPC2. The suboptimal result on "Hopper Hop" is examined through the latent task embeddings below.

Task-Embedding Failure Analysis. Figure 19 below shows the task embedding $\mathcal{E}_{env}$ for 70-task pretraining with 10 unseen tasks for fine-tuning. Each point corresponds to one task embedding projected into a two-dimensional space using t-SNE for visualization. The red circles denote MetaWorld tasks, while the blue squares denote DMControl tasks. As shown in the figure, the learned task embeddings exhibit a clear domain-level separation: MetaWorld manipulation tasks form a compact cluster in the upper region, whereas DMControl locomotion and control tasks are mainly grouped in the lower region. This separation indicates that $\mathcal{E}_{env}$ captures task-specific environment information and organizes tasks according to their underlying dynamics, observation distributions, and action semantics.

Most tasks within the same benchmark are embedded close to semantically or dynamically related tasks. For example, MetaWorld tasks involving similar manipulation primitives, such as pushing, pulling, opening, closing, and object placement, tend to be located near each other. Similarly, DMControl tasks with related morphology or control structure, such as Walker, Cheetah, Cartpole, and Reacher variants, also form local clusters. This suggests that the learned task embedding is not merely distinguishing task identities, but also reflects meaningful similarities across tasks. Notably, the task "Hopper Hop" is highlighted by the red circle. Although it belongs to the DMControl suite, its embedding is relatively isolated from the main DMControl cluster. This indicates that "Hopper Hop" has a substantially different data distribution from the other pretraining tasks. Such an outlying embedding may lead to less effective sharing of representations and policies during multi-task pretraining, thereby making policy optimization more challenging for this task. Overall, the visualization supports our hypothesis that the learned task embedding $\mathcal{E}_{env}$ provides a structured representation of task distributions, while also revealing tasks whose distributions deviate significantly from the majority of the pretraining set.



Figure 19: **T-SNE visualization (Van der Maaten & Hinton, 2008) of task embeddings learned by MBDPO trained on** 80 **tasks from DMControl and MetaWorld.** Detailed labels are shown for clarity. According to our hypothesis, the task embedding of Hopper Hop can be attributed to a severe deviation in the representation's data distribution (see the red circle), which subsequently impedes effective policy optimization.

G.2.5 Additional Ablations

Figures 20–22 provide detailed results for the energy weight η, Monte Carlo sample count T, and diffusion denoising steps N summarized in Figure 6.

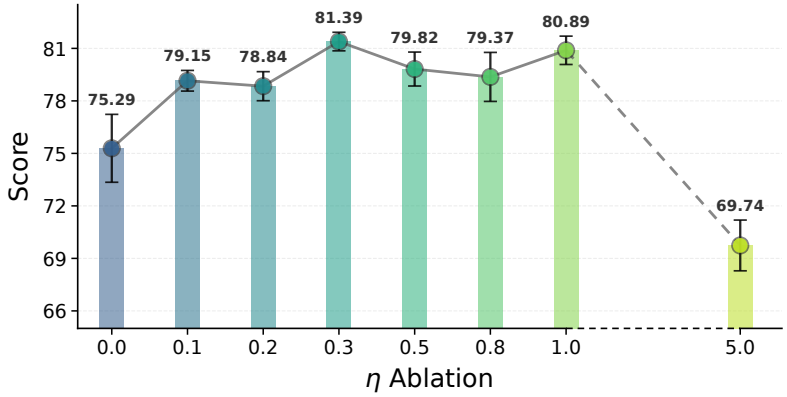

Figure 20: **Effect of the energy weight η.** Results use the 80-task dataset and a 21M-parameter model. Performance remains stable under moderate regularization but degrades when η becomes too large.

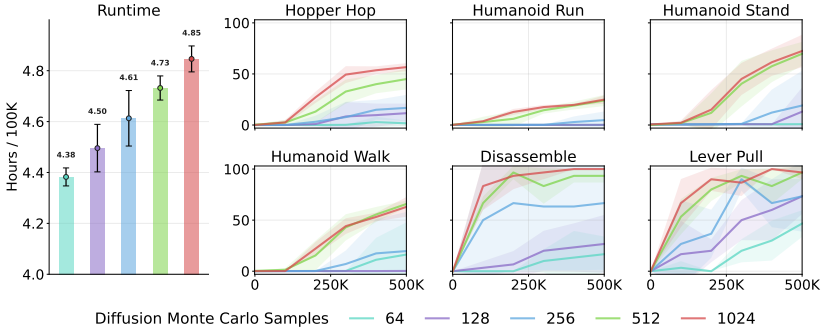

Figure 21: **Effect of Monte Carlo samples T.** Increasing T improves returns across representative tasks with only a modest increase in training time. Results are averaged over three seeds.

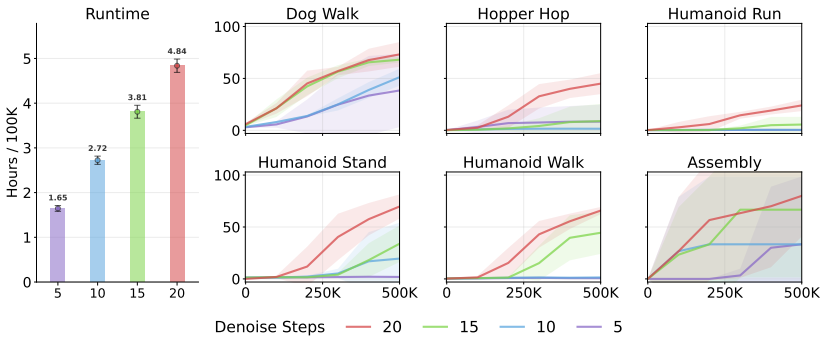

Figure 22: **Effect of diffusion denoising steps N.** Increasing N improves returns but also increases runtime, showing a direct performance–computation trade-off. Results are averaged over three seeds.

G.2.6 Representation Analysis

Latent Trajectories Comparison. We provide additional visualizations of latent state trajectories for all 80 tasks under both MBDPO and TD-MPC2. Specifically, Figures 24 and 25 compare single-episode trajectories, while Figures 26 and 27 compare multi-episode trajectories. Across both settings, MBDPO produces more structured and temporally coherent latent manifolds, including closed-loop patterns for cyclic control tasks and smooth goal-directed transitions for manipulation tasks. In contrast, TD-MPC2 often exhibits noisier and less organized trajectories, suggesting weaker alignment between the sampled policy and the underlying task dynamics.

These visualizations provide qualitative evidence that MBDPO produces more coherent latent trajectories than TD-MPC2. Because the plots rely on dimensionality reduction, we use them only as supporting observations rather than as direct evidence of causal or physical structure.

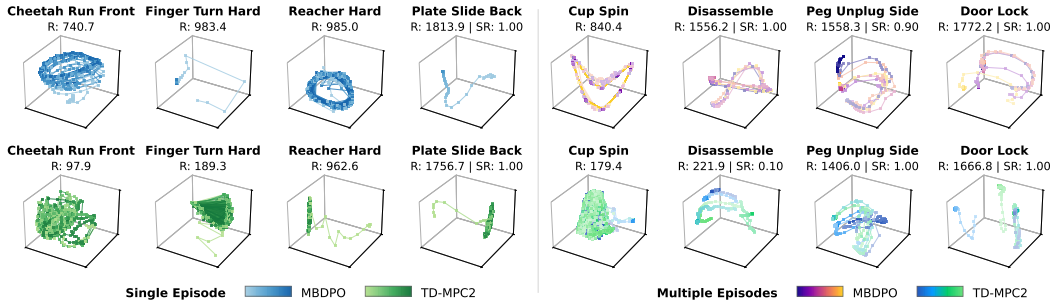

Figure 23: **Latent trajectory structure.** Across single- and multi-episode rollouts, MBDPO produces coherent closed loops for cyclic tasks and smooth goal-directed paths for manipulation tasks, while TD-MPC2 yields noisier trajectories. Colors indicate temporal progression, while R and SR denote return and success rate. Results for all 80 tasks are provided in Appendix Figures 24–27.

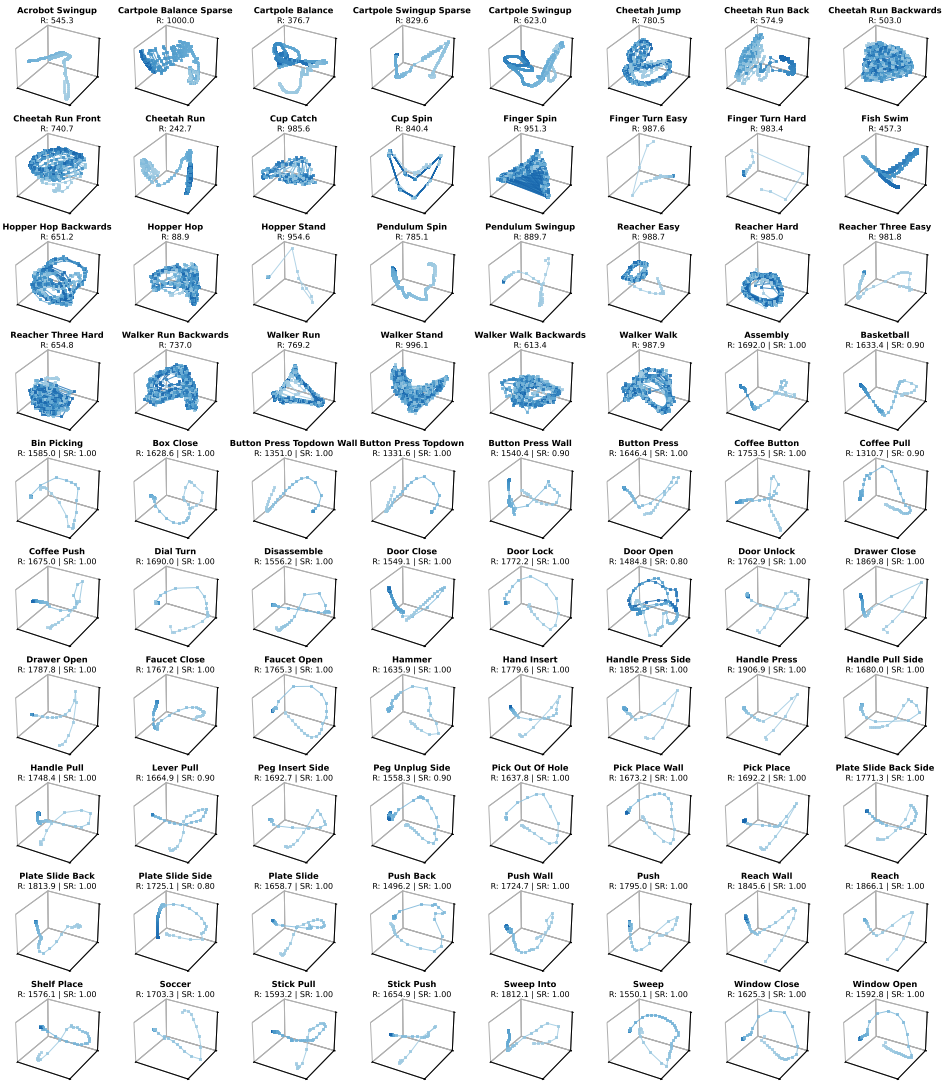

Figure 24: **Single-episode latent trajectory visualization for MBDPO.** We visualize the latent state trajectories produced by MBDPO with the diffusion policy over one episode for all 80 tasks using a locally linear embedding. The color gradient indicates temporal progression along the trajectory. Cyclical control tasks often form closed-loop patterns, while goal-directed manipulation tasks typically evolve smoothly from the initial state toward the target state.

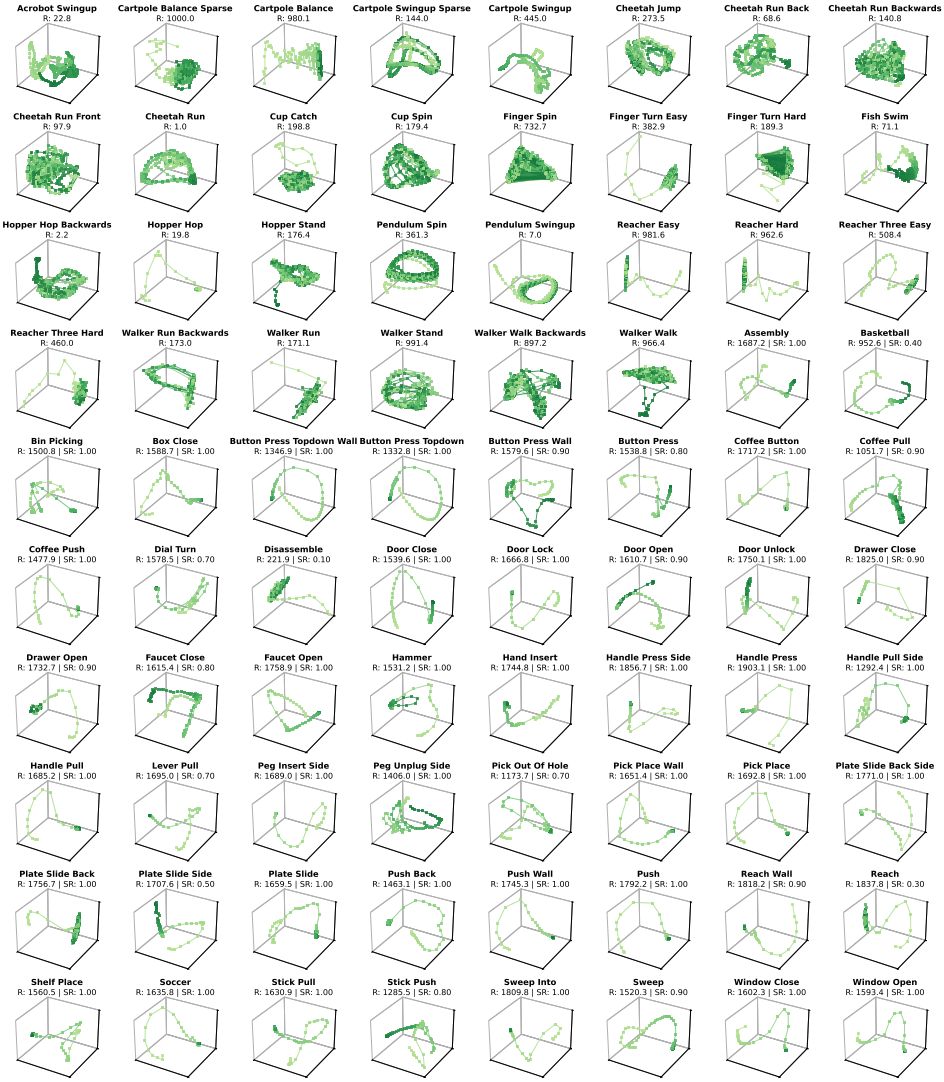

Figure 25: **Single-episode latent trajectory visualization for TD-MPC2.** We visualize the latent state trajectories produced by TD-MPC2 over one episode for all 80 tasks using the same locally linear embedding protocol. The color gradient indicates temporal progression along the trajectory. Compared with MBDPO, TD-MPC2 produces trajectories that are often more scattered, noisy, and less organized under the same embedding protocol.

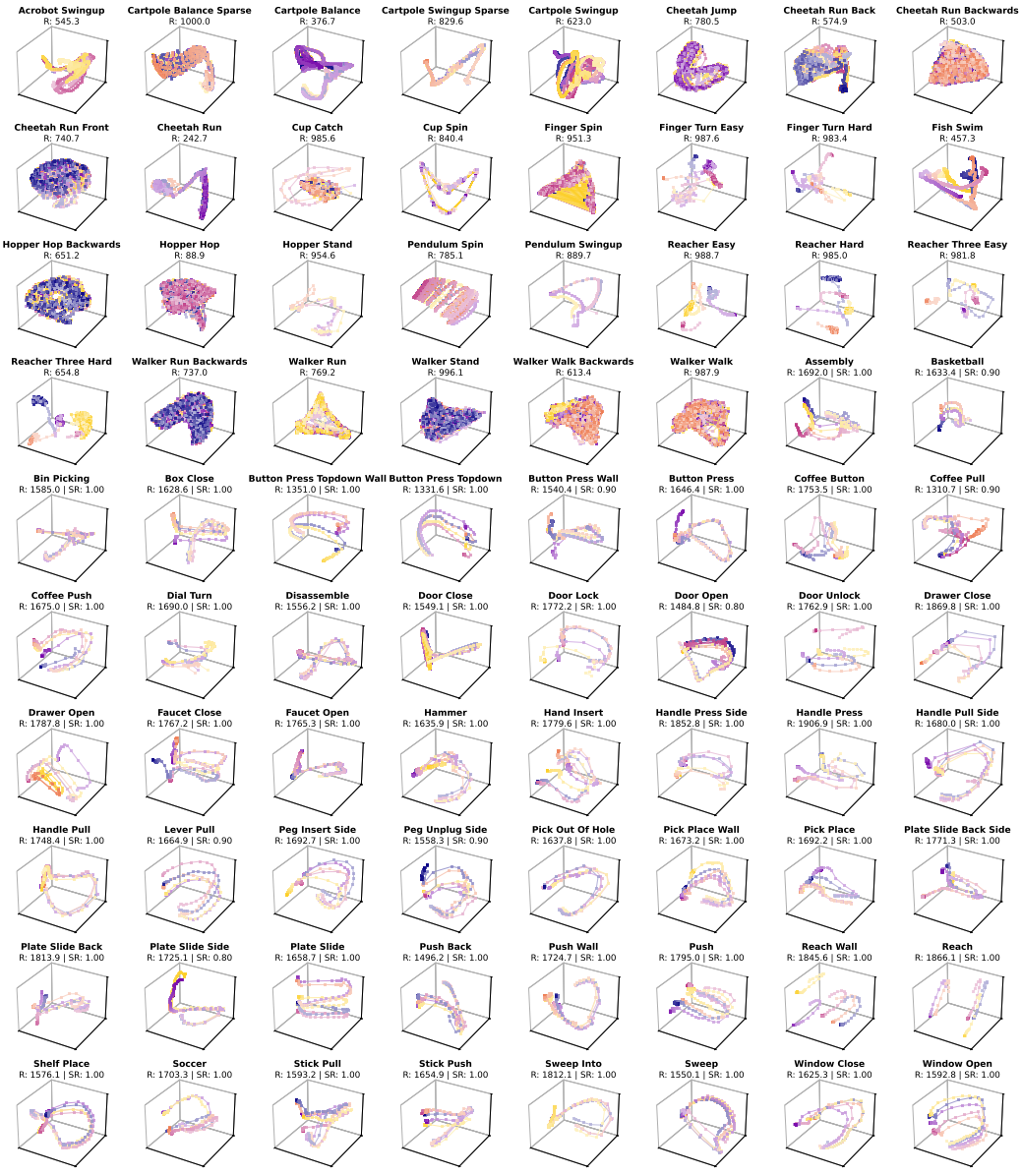

Figure 26: **Multi-episode latent trajectory visualization for MBDPO.** We visualize latent state trajectories produced by MBDPO with the diffusion policy over multiple episodes for all 80 tasks using a locally linear embedding. The color gradient denotes temporal progression within each episode. Across repeated rollouts, MBDPO maintains consistent and structured trajectory manifolds, with closed-loop patterns for cyclic tasks and smooth directed transitions for goal-conditioned manipulation tasks. These results indicate that the diffusion policy captures stable task-specific latent dynamics across episodes.

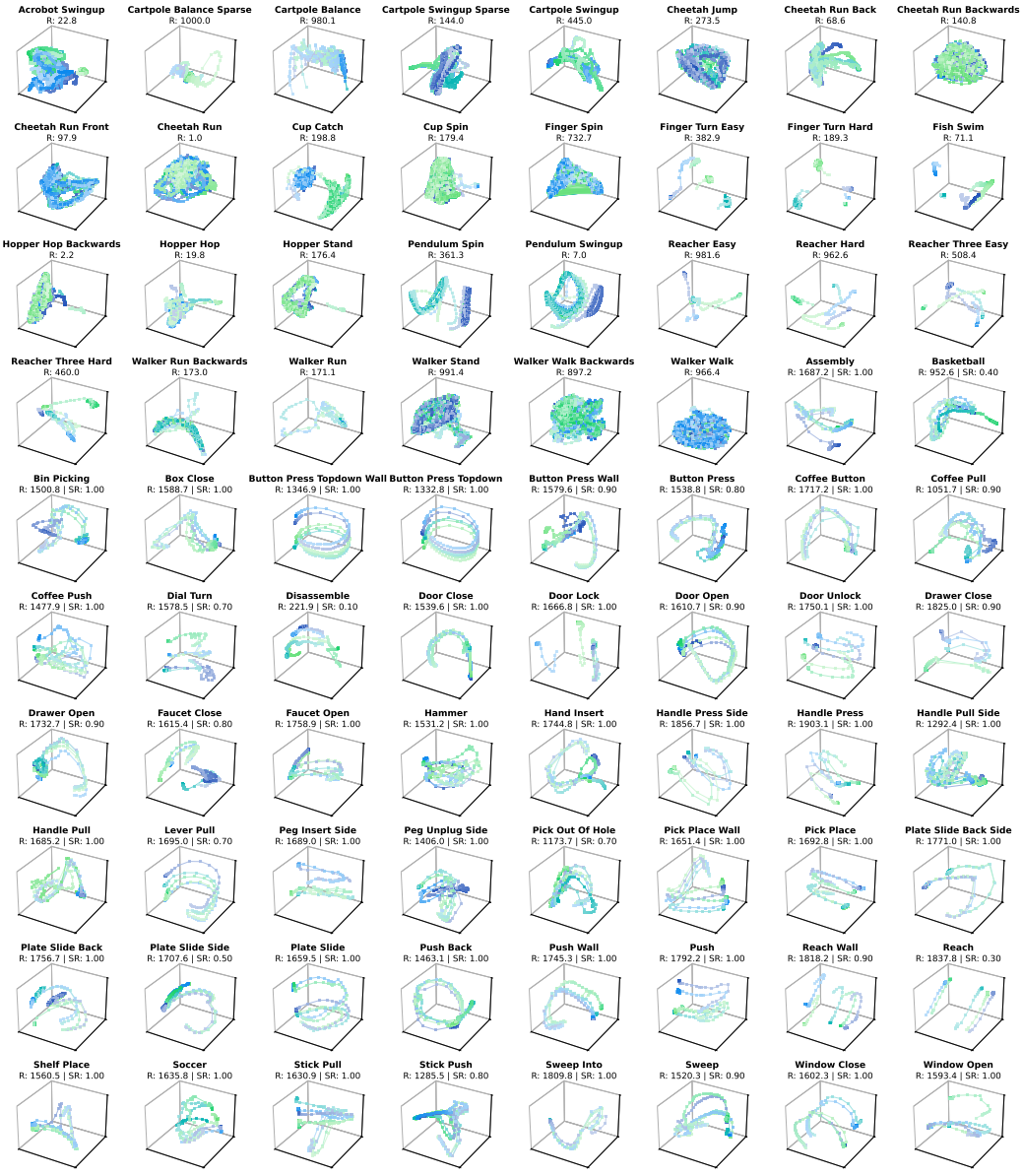

Figure 27: **Multi-episode latent trajectory visualization for TD-MPC2.** We visualize latent state trajectories produced by TD-MPC2 over multiple episodes for all 80 tasks using the same locally linear embedding protocol. The color gradient denotes temporal progression within each episode. Compared with MBDPO, TD-MPC2 exhibits less consistent trajectory geometry across rollouts, with noisier, more dispersed, and more irregular latent paths.

G.2.7 COMPUTATIONAL COST

MBDPO incurs negligible overhead in online-from-scratch training. Across 121 tasks from five domains, its per-step training time is comparable to TD-MPC2, and is faster on DMControl, MyoSuite, and Visual RL, reducing the Visual RL average from 3.55 ms to 3.04 ms. For multi-task offline training, MBDPO scales efficiently from 30 to 80 tasks, with only a $1.02\times$–$1.32\times$ increase in total training time. Notably, the 340M model shows nearly unchanged cost (202.9 vs. 206.8 hours), indicating strong scalability to broader multi-task settings. Table 1 summarizes these results, followed by the task-level measurements.

Table 1: **Profiling training efficiency results for online-from-scratch and multi-task offline pretraining.** (a) Online training time is measured per environment step (milliseconds) on a single NVIDIA A800-80GB, see the task-specific results in Table 2. (b) Offline training time is reported in hours for 10M training steps on a single NVIDIA-H200. Values are reported as mean $\pm$ std.

(a) Online-from-scratch training time

Domain	#	TD-MPC2	MBDPO
MetaWorld	50	1.08 ± 0.16	1.08 ± 0.15
DMControl	39	0.78 ± 0.71	0.76 ± 0.71
ManiSkill2	5	6.50 ± 1.83	6.91 ± 2.51
MyoSuite	10	1.58 ± 0.22	1.54 ± 0.19
Visual RL	10	3.55 ± 0.51	3.04 ± 0.37

(b) Multi-task offline training time

Parameters	30 tasks	80 tasks	80/30
1.7M	34.8 ± 0.6	39.9 ± 0.4	$1.15\times$
6M	35.0 ± 0.5	40.5 ± 0.6	$1.16\times$
21M	39.8 ± 0.3	42.6 ± 0.4	$1.07\times$
54M	52.6 ± 0.1	69.6 ± 0.7	$1.32\times$
340M	202.9 ± 0.9	206.8 ± 0.8	$1.02\times$

Table 2: **Profiling results of online-task training time per environment step across domains** (milliseconds per env step). Values are reported as mean ± std. Each experiment runs a single task on one NVIDIA A800-SXM4-80GB GPU.

Task	TD-MPC2	MBDPO	Task	TD-MPC2	MBDPO
MetaWorld					
mw-assembly	1.35 ± 0.012	1.35 ± 0.008	mw-basketball	1.07 ± 0.019	1.09 ± 0.022
mw-button-press-topdown	0.87 ± 0.007	0.89 ± 0.006	mw-button-press-topdown-wall	0.90 ± 0.016	0.89 ± 0.006
mw-button-press	0.89 ± 0.017	0.92 ± 0.028	mw-button-press-wall	0.94 ± 0.054	0.91 ± 0.013
mw-coffee-button	0.99 ± 0.015	1.00 ± 0.004	mw-coffee-pull	1.26 ± 0.026	1.26 ± 0.007
mw-coffee-push	1.22 ± 0.007	1.21 ± 0.006	mw-dial-turn	0.94 ± 0.024	0.92 ± 0.007
mw-disassemble	1.36 ± 0.018	1.34 ± 0.020	mw-door-open	1.09 ± 0.013	1.08 ± 0.015
mw-door-close	1.01 ± 0.010	1.01 ± 0.016	mw-drawer-close	1.09 ± 0.041	1.08 ± 0.041
mw-drawer-open	1.02 ± 0.007	1.03 ± 0.012	mw-faucet-open	0.91 ± 0.022	0.90 ± 0.029
mw-faucet-close	0.90 ± 0.005	0.89 ± 0.007	mw-hammer	1.33 ± 0.016	1.33 ± 0.016
mw-handle-press-side	0.88 ± 0.035	0.87 ± 0.018	mw-handle-press	0.93 ± 0.038	0.92 ± 0.037
mw-handle-pull-side	1.00 ± 0.011	1.00 ± 0.036	mw-handle-pull	0.99 ± 0.009	0.98 ± 0.007
mw-lever-pull	1.00 ± 0.014	0.99 ± 0.010	mw-peg-insert-side	1.14 ± 0.005	1.14 ± 0.016
mw-peg-unplug-side	1.22 ± 0.079	1.20 ± 0.079	mw-pick-out-of-hole	1.08 ± 0.019	1.07 ± 0.048
mw-pick-place	1.07 ± 0.014	1.10 ± 0.032	mw-pick-place-wall	1.19 ± 0.010	1.18 ± 0.006
mw-plate-slide	0.90 ± 0.011	0.92 ± 0.013	mw-plate-slide-side	0.92 ± 0.008	0.91 ± 0.013
mw-plate-slide-back	0.97 ± 0.011	0.94 ± 0.013	mw-plate-slide-back-side	0.96 ± 0.011	0.95 ± 0.009
mw-push-back	1.10 ± 0.008	1.10 ± 0.011	mw-push	1.11 ± 0.010	1.11 ± 0.008
mw-push-wall	1.17 ± 0.010	1.15 ± 0.014	mw-reach	0.90 ± 0.011	0.89 ± 0.011
mw-reach-wall	0.91 ± 0.005	0.92 ± 0.025	mw-shelf-place	1.21 ± 0.040	1.19 ± 0.038
mw-soccer	1.10 ± 0.014	1.10 ± 0.009	mw-stick-push	1.35 ± 0.015	1.34 ± 0.010
mw-stick-pull	1.43 ± 0.017	1.39 ± 0.009	mw-sweep-into	1.19 ± 0.030	1.20 ± 0.023
mw-sweep	1.12 ± 0.024	1.11 ± 0.004	mw-window-open	1.23 ± 0.018	1.22 ± 0.020
mw-window-close	1.24 ± 0.031	1.23 ± 0.008	mw-bin-picking	1.12 ± 0.015	1.19 ± 0.229
mw-box-close	1.44 ± 0.092	1.41 ± 0.087	mw-door-lock	1.02 ± 0.067	1.00 ± 0.076
mw-door-unlock	1.00 ± 0.042	1.00 ± 0.041	mw-hand-insert	1.14 ± 0.031	1.11 ± 0.026
DMControl					
acrobot-swingup	0.40 ± 0.003	0.38 ± 0.002	cartpole-balance	0.51 ± 0.002	0.50 ± 0.012
cartpole-balance-sparse	0.40 ± 0.004	0.37 ± 0.002	cartpole-swingup	0.51 ± 0.013	0.51 ± 0.001
cartpole-swingup-sparse	0.38 ± 0.002	0.37 ± 0.001	cheetah-jump	0.50 ± 0.007	0.50 ± 0.009
cheetah-run	0.40 ± 0.005	0.38 ± 0.008	cheetah-run-back	0.45 ± 0.003	0.45 ± 0.008
cheetah-run-backwards	0.40 ± 0.008	0.38 ± 0.003	cheetah-run-front	0.47 ± 0.007	0.44 ± 0.004
cup-catch	0.40 ± 0.011	0.39 ± 0.002	cup-spin	0.68 ± 0.055	0.67 ± 0.050
dog-run	2.76 ± 0.147	2.70 ± 0.137	dog-trot	2.72 ± 0.097	2.84 ± 0.118
dog-stand	2.69 ± 0.069	2.62 ± 0.090	dog-walk	2.85 ± 0.151	2.74 ± 0.138
finger-spin	0.37 ± 0.009	0.33 ± 0.002	finger-turn-easy	0.47 ± 0.005	0.45 ± 0.003
finger-turn-hard	0.47 ± 0.005	0.46 ± 0.005	fish-swim	0.51 ± 0.003	0.51 ± 0.003
hopper-hop	0.45 ± 0.019	0.41 ± 0.015	hopper-hop-backwards	0.53 ± 0.043	0.41 ± 0.016
hopper-stand	0.49 ± 0.014	0.46 ± 0.015	humanoid-run	1.03 ± 0.013	1.02 ± 0.012
humanoid-stand	1.07 ± 0.010	1.09 ± 0.009	humanoid-walk	1.06 ± 0.011	1.03 ± 0.012
pendulum-spin	0.35 ± 0.011	0.33 ± 0.001	pendulum-swingup	0.31 ± 0.002	0.30 ± 0.001
quadruped-run	1.11 ± 0.020	1.09 ± 0.019	quadruped-walk	1.09 ± 0.019	1.09 ± 0.022
reacher-easy	0.37 ± 0.003	0.34 ± 0.004	reacher-hard	0.36 ± 0.003	0.34 ± 0.002
reacher-three-easy	0.38 ± 0.007	0.36 ± 0.003	reacher-three-hard	0.38 ± 0.006	0.35 ± 0.003
walker-run	0.61 ± 0.006	0.58 ± 0.006	walker-run-backwards	0.61 ± 0.005	0.59 ± 0.007
walker-stand	0.57 ± 0.010	0.55 ± 0.021	walker-walk	0.61 ± 0.006	0.58 ± 0.003
walker-walk-backwards	0.60 ± 0.007	0.59 ± 0.007			
ManiSkill2					
lift-cube	4.98 ± 0.301	4.85 ± 0.249	pick-cube	4.84 ± 0.389	5.04 ± 0.251
stack-cube	5.74 ± 0.229	5.85 ± 0.351	pick-ycb	8.24 ± 1.921	7.99 ± 1.890
turn-faucet	8.69 ± 2.773	10.81 ± 5.652			
MyoSuite					
myo-reach	1.59 ± 0.062	1.52 ± 0.029	myo-reach-hard	1.59 ± 0.041	1.55 ± 0.024
myo-pose	1.49 ± 0.039	1.48 ± 0.049	myo-pose-hard	1.65 ± 0.046	1.59 ± 0.087
myo-obj-hold	1.68 ± 0.058	1.61 ± 0.036	myo-obj-hold-hard	1.58 ± 0.043	1.61 ± 0.040
myo-key-turn	1.84 ± 0.081	1.70 ± 0.085	myo-key-turn-hard	1.90 ± 0.097	1.86 ± 0.090
myo-pen-twirl	1.23 ± 0.022	1.22 ± 0.039	myo-pen-twirl-hard	1.24 ± 0.037	1.24 ± 0.019
Visual RL					
acrobot-swingup	2.99 ± 0.023	2.54 ± 0.070	cheetah-run	3.20 ± 0.153	3.16 ± 0.553
finger-spin	4.18 ± 0.212	2.98 ± 0.438	finger-turn-easy	4.10 ± 0.086	2.96 ± 0.050
finger-turn-hard	3.38 ± 0.279	3.04 ± 0.350	quadruped-walk	4.42 ± 0.101	3.72 ± 0.029
reacher-easy	3.48 ± 0.280	2.80 ± 0.076	reacher-hard	2.91 ± 0.027	2.53 ± 0.011
walker-run	3.45 ± 0.104	3.43 ± 0.439	walker-walk	3.44 ± 0.089	3.24 ± 0.048

Table 3: **Profiling results of evaluation time for online tasks** (seconds per evaluation phase, 10 episodes in total) across domains. Values are reported as mean ± std. All experiments are evaluated with a single task running on a single NVIDIA A800-SXM4-80GB GPU.

Task	TD-MPC2	MBDPO	Task	TD-MPC2	MBDPO
MetaWorld					
mw-assembly	10.67 ± 0.03	18.89 ± 0.02	mw-basketball	10.31 ± 0.01	18.76 ± 0.04
mw-button-press-topdown	10.18 ± 0.01	18.77 ± 0.02	mw-button-press-topdown-wall	10.32 ± 0.01	18.73 ± 0.08
mw-button-press	10.06 ± 0.01	18.39 ± 0.02	mw-button-press-wall	10.16 ± 0.01	18.51 ± 0.02
mw-coffee-button	10.33 ± 0.02	18.48 ± 0.03	mw-coffee-pull	10.59 ± 0.04	18.73 ± 0.16
mw-coffee-push	10.44 ± 0.01	18.79 ± 0.02	mw-dial-turn	10.25 ± 0.01	18.31 ± 0.03
mw-disassemble	10.48 ± 0.01	18.68 ± 0.09	mw-door-open	10.29 ± 0.01	18.69 ± 0.01
mw-door-close	10.26 ± 0.01	19.02 ± 0.04	mw-drawer-close	10.36 ± 0.03	18.60 ± 0.02
mw-drawer-open	10.32 ± 0.05	18.67 ± 0.08	mw-faucet-open	10.36 ± 0.06	18.38 ± 0.12
mw-faucet-close	10.14 ± 0.02	18.15 ± 0.04	mw-hammer	10.62 ± 0.03	18.87 ± 0.04
mw-handle-press-side	10.08 ± 0.00	18.53 ± 0.09	mw-handle-press	10.15 ± 0.02	18.61 ± 0.04
mw-handle-pull-side	10.21 ± 0.00	18.40 ± 0.02	mw-handle-pull	10.25 ± 0.03	18.52 ± 0.04
mw-lever-pull	10.31 ± 0.02	18.73 ± 0.12	mw-peg-insert-side	10.38 ± 0.05	18.79 ± 0.04
mw-peg-unplug-side	10.42 ± 0.03	18.73 ± 0.04	mw-pick-out-of-hole	10.66 ± 0.01	18.73 ± 0.02
mw-pick-place	10.33 ± 0.01	18.50 ± 0.07	mw-pick-place-wall	10.45 ± 0.02	18.70 ± 0.03
mw-plate-slide	10.27 ± 0.01	18.42 ± 0.03	mw-plate-slide-side	10.15 ± 0.01	18.57 ± 0.30
mw-plate-slide-back	10.36 ± 0.04	18.67 ± 0.03	mw-plate-slide-back-side	10.42 ± 0.02	19.25 ± 0.09
mw-push-back	10.32 ± 0.01	19.04 ± 0.03	mw-push	10.37 ± 0.02	18.64 ± 0.04
mw-push-wall	10.62 ± 0.05	18.62 ± 0.02	mw-reach	10.16 ± 0.01	18.60 ± 0.04
mw-reach-wall	10.14 ± 0.03	18.42 ± 0.04	mw-shelf-place	10.42 ± 0.02	18.64 ± 0.02
mw-soccer	10.45 ± 0.05	19.08 ± 0.05	mw-stick-push	10.57 ± 0.05	19.04 ± 0.03
mw-stick-pull	10.65 ± 0.02	19.19 ± 0.03	mw-sweep-into	10.64 ± 0.05	18.53 ± 0.03
mw-sweep	10.32 ± 0.01	18.71 ± 0.02	mw-window-open	10.48 ± 0.02	18.48 ± 0.02
mw-window-close	10.41 ± 0.01	18.59 ± 0.03	mw-bin-picking	10.36 ± 0.01	19.01 ± 0.03
mw-box-close	10.51 ± 0.02	18.44 ± 0.08	mw-door-lock	10.47 ± 0.05	18.65 ± 0.06
mw-door-unlock	10.45 ± 0.02	18.86 ± 0.04	mw-hand-insert	10.73 ± 0.04	18.89 ± 0.04
DMControl					
acrobot-swingup	48.96 ± 0.04	91.41 ± 0.05	cartpole-balance	49.54 ± 0.02	93.11 ± 0.49
cartpole-balance-sparse	48.59 ± 0.02	89.92 ± 0.31	cartpole-swingup	49.14 ± 0.02	95.71 ± 0.21
cartpole-swingup-sparse	48.53 ± 0.02	90.39 ± 0.64	cheetah-jump	48.64 ± 0.02	90.12 ± 0.42
cheetah-run	48.04 ± 0.05	88.86 ± 0.06	cheetah-run-back	48.40 ± 0.07	89.86 ± 0.05
cheetah-run-backwards	48.28 ± 0.19	89.26 ± 0.20	cheetah-run-front	48.57 ± 0.18	93.52 ± 0.31
cup-catch	48.53 ± 0.06	89.35 ± 0.33	cup-spin	49.28 ± 0.08	88.98 ± 0.08
dog-run	81.49 ± 0.49	116.13 ± 0.88	dog-trot	82.57 ± 0.43	112.60 ± 0.74
dog-stand	80.86 ± 0.47	117.07 ± 0.78	dog-walk	76.16 ± 0.14	102.96 ± 0.42
finger-spin	48.39 ± 0.08	88.62 ± 0.51	finger-turn-easy	48.70 ± 0.07	90.84 ± 0.38
finger-turn-hard	49.44 ± 0.47	90.07 ± 0.18	fish-swim	50.22 ± 0.40	90.99 ± 0.75
hopper-hop	48.45 ± 0.05	90.15 ± 0.08	hopper-hop-backwards	48.56 ± 0.03	90.67 ± 0.06
hopper-stand	48.93 ± 0.04	89.40 ± 0.07	humanoid-run	65.93 ± 0.16	93.01 ± 0.31
humanoid-stand	66.68 ± 0.42	93.37 ± 0.98	humanoid-walk	65.95 ± 0.19	93.96 ± 0.15
pendulum-spin	48.23 ± 0.04	89.49 ± 0.40	pendulum-swingup	48.29 ± 0.09	89.57 ± 0.93
quadruped-run	51.07 ± 0.10	92.25 ± 0.34	quadruped-walk	50.48 ± 0.06	94.07 ± 0.14
reacher-easy	48.58 ± 0.05	89.32 ± 0.39	reacher-hard	48.50 ± 0.19	89.07 ± 0.38
reacher-three-easy	48.62 ± 0.04	89.35 ± 0.04	reacher-three-hard	48.29 ± 0.08	88.68 ± 0.42
walker-run	49.95 ± 0.03	90.58 ± 0.17	walker-run-backwards	49.51 ± 0.03	89.00 ± 0.12
walker-stand	49.05 ± 0.03	92.53 ± 0.13	walker-walk	49.31 ± 0.08	90.07 ± 0.08
walker-walk-backwards	49.61 ± 0.04	90.30 ± 0.06			
ManiSkill2					
lift-cube	15.27 ± 0.32	24.99 ± 1.19	pick-cube	15.62 ± 0.27	26.34 ± 0.76
stack-cube	16.08 ± 0.30	27.02 ± 0.65	pick-ycb	18.83 ± 1.01	31.01 ± 2.20
turn-faucet	18.84 ± 1.11	25.06 ± 0.55			
MyoSuite					
myo-reach	13.90 ± 0.03	19.22 ± 0.02	myo-reach-hard	14.00 ± 0.04	19.45 ± 0.09
myo-pose	13.90 ± 0.07	19.00 ± 0.02	myo-pose-hard	13.95 ± 0.04	19.08 ± 0.03
myo-obj-hold	13.88 ± 0.05	19.79 ± 0.02	myo-obj-hold-hard	14.01 ± 0.05	19.34 ± 0.05
myo-key-turn	14.27 ± 0.08	18.84 ± 0.04	myo-key-turn-hard	14.10 ± 0.03	19.22 ± 0.05
myo-pen-twirl	13.66 ± 0.05	18.81 ± 0.03	myo-pen-twirl-hard	13.67 ± 0.07	18.72 ± 0.02
Visual RL					
acrobot-swingup	66.51 ± 0.42	108.95 ± 0.30	cheetah-run	66.29 ± 0.37	108.64 ± 0.31
finger-spin	66.35 ± 0.47	108.86 ± 0.28	finger-turn-easy	66.36 ± 0.46	113.39 ± 0.09
finger-turn-hard	65.37 ± 2.59	105.12 ± 0.20	quadruped-walk	66.66 ± 1.69	106.39 ± 0.67
reacher-easy	65.29 ± 2.59	104.03 ± 0.94	reacher-hard	65.41 ± 2.63	104.97 ± 0.48
walker-run	66.02 ± 0.15	104.89 ± 0.63	walker-walk	66.13 ± 0.23	105.64 ± 0.20

G.3 EXPERIMENTS ON MMBENCH

MMBench overview. MMBench is a massively multi-task continuous-control benchmark comprising 200 training tasks from ten domains: DMControl (21), DMControl Extended (16), Meta-World (49), ManiSkill3 (36), MuJoCo (6), Box2D (8), RoboDesk (6), OGBench (12), MiniArcade (19), and Atari (27). The collection combines 159 existing tasks, 22 newly introduced variants in established domains, and 19 new MiniArcade tasks, spanning locomotion, robotic manipulation, navigation, classical control, and arcade games. Its tasks differ substantially in embodiment, observation and action dimensionality, episode length (25–1,000 steps), and reward structure. Every task provides a manually specified language instruction describing both the embodiment/action space and the task objective, low-dimensional state observations, optional 224×224 RGB observations, and expert demonstrations. The released dataset contains 4,020 demonstrations—10–40 episodes per task—collected using converged single-task TD-MPC2 agents. This combination is designed to test whether one shared agent can disambiguate tasks, transfer representations and behavioral knowledge across heterogeneous domains, exploit demonstrations for exploration, and continue improving through online interaction rather than merely fitting an offline dataset.

Online training and evaluation protocol. We follow the original state-based MMBench protocol and jointly train a single language-conditioned agent on all 200 tasks for 100M aggregate environment transitions. The implementation creates one asynchronous environment per task; under multi-GPU training, tasks are partitioned round-robin across workers while gradients are synchronized, so each task receives approximately 500K online transitions. Heterogeneous state and action spaces are padded to shared 128- and 16-dimensional representations, respectively, while masks prevent invalid action dimensions from affecting inference or learning. Each task is identified by a cached 512-dimensional CLIP embedding of its instruction, and task-dependent episode lengths and discount factors are retained. Before online interaction, the complete world model and policy prior are pretrained for 200K updates on demonstrations. During online learning, complete episodes generated by the planner are added to a 10M-transition replay buffer, and each update uniformly samples short trajectory subsequences, with demonstrations and online experience contributing equal halves of the batch. In our MBDPO implementation, diffusion-based latent trajectory optimization is executed in a receding-horizon manner, with only the first optimized action applied before replanning. The reported environment-step count sums transitions over all tasks. For evaluation, each task is evaluated independently, and its return or success is converted by the corresponding environment wrapper to a score in $[0, 1]$. Scores are first averaged across evaluation episodes for each task and then averaged equally across all 200 tasks; domain-level results analogously average only the tasks belonging to that domain. Thus, the aggregate metric assigns equal weight to tasks rather than equal weight to domains.

Experimental results and analysis. Under this protocol, MBDPO substantially improves massively multi-task online learning. At 100M aggregate environment steps, the 366.7M-parameter configuration obtains an overall normalized score of 0.490, compared with 0.438 for Newt, 0.273 for FastTD3, 0.262 for TD-MPC2, 0.137 for PPO, and 0.235 for the collection of single-task BC policies. MBDPO achieves the strongest result among the compared methods in eight of the ten domains and exceeds Newt in nine domains, with especially clear gains on Meta-World (0.624 versus 0.503), OGBench (0.379 versus 0.304), RoboDesk (0.450 versus 0.375), and DMControl (0.561 versus 0.505); Atari remains the sole domain on which Newt is slightly stronger. The advantage is also present at approximately matched model capacities: averaged over the final 10% of training, the 22.5M-parameter MBDPO model scores 0.447 versus 0.437 for 20M Newt, while 88.3M MBDPO reaches 0.465 versus 0.447 for 80M Newt. Further scaling raises MBDPO's score to 0.487 at 366.7M parameters and 0.496 at 1.82B parameters. These results show that the improvement is not solely due to comparing unequal capacities: MBDPO improves performance at matched scales and continues to benefit from additional capacity, demonstrating effective cross-domain knowledge sharing and stable scaling of a single online world-model agent across highly heterogeneous tasks.

G.3.1 EVALUATION SETUP

We further evaluate MBDPO on MMBench, a 200-task benchmark covering ten domains: DMControl, DMControl Extended, Meta-World, ManiSkill, MuJoCo, Box2D, RoboDesk, OGBench, MiniArcade,

and Atari. We report overall average scaling trends, domain-level comparisons with Newt and standard RL baselines, and complete MBDPO learning curves for every task.

MBDPO follows the data and conditioning protocol used by Newt. Both methods use the same task demonstrations, precomputed and frozen 512-dimensional task-language embeddings, and an online interaction budget of 100M aggregate environment steps. The primary difference is how policy optimization is performed within the world model.

G.3.2 Scaling and Domain-Level Performance

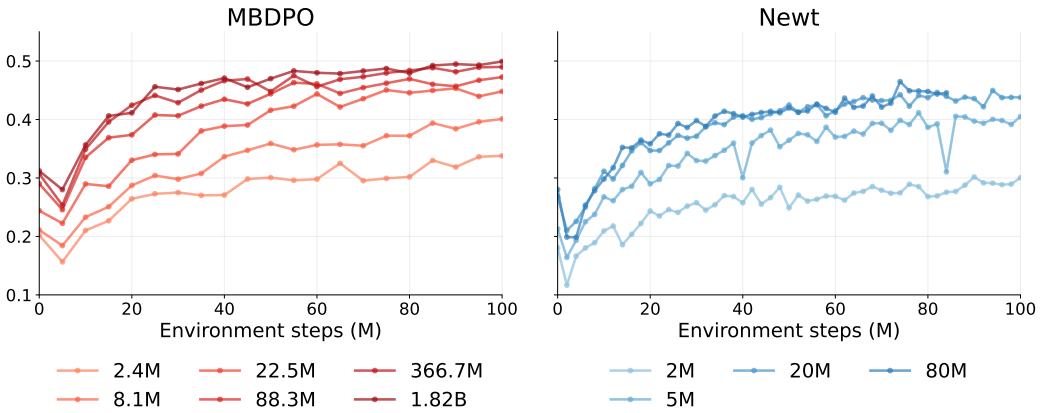

Figure 28: **Scaling performance on MMBench.** Average score when training a single agent via online interaction on 200 tasks spanning 10 task domains. The left panel reports results from MBDPO models with 2.4M, 8.1M, 22.5M, 88.3M, 366.7M, and 1.82B parameters, while the right panel reports results from Newt models with 2M, 5M, 20M, and 80M parameters. MBDPO exhibits a clear scaling trend, with larger models generally achieving better scaling performance throughout training.

G.3.3 Per-Task Results

Figures 29–38 provide the complete task-level learning curves used to compute the aggregate results above. All curves report per-task normalized scores over a 100M-step training horizon, where environment steps denote the total interaction steps accumulated across all tasks.

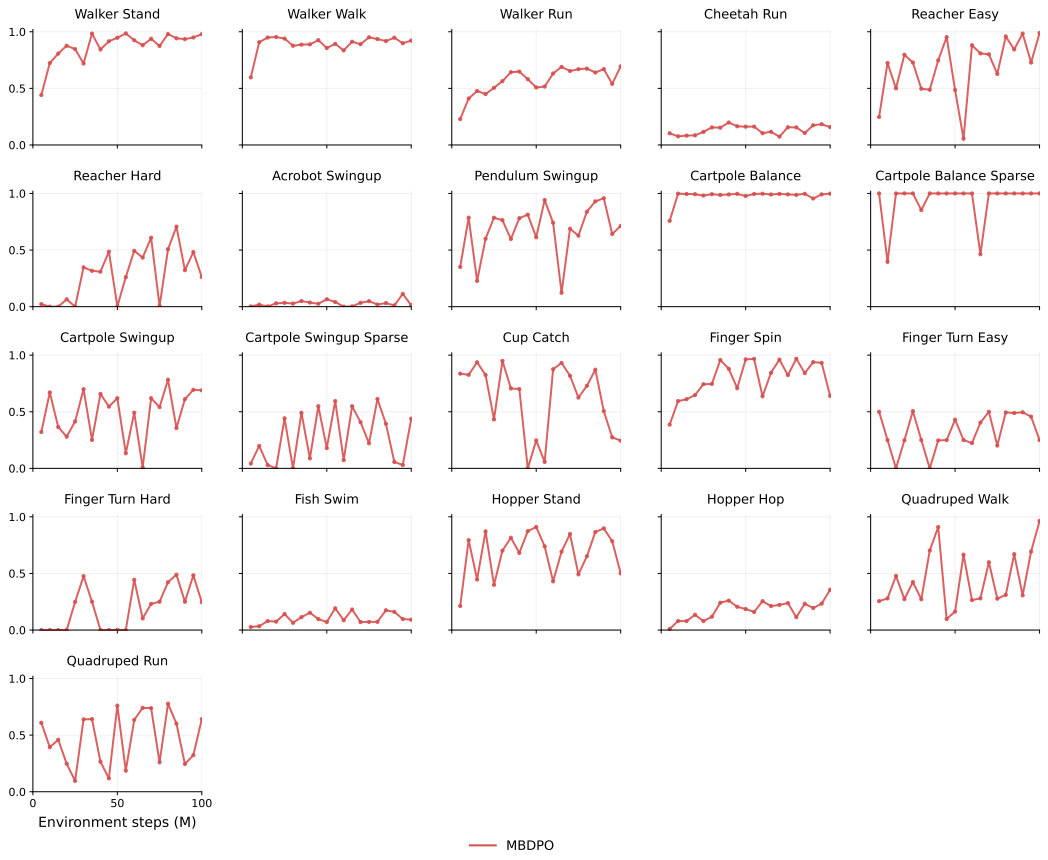

Figure 29: **DMControl.** Per-task normalized scores for 21 tasks.

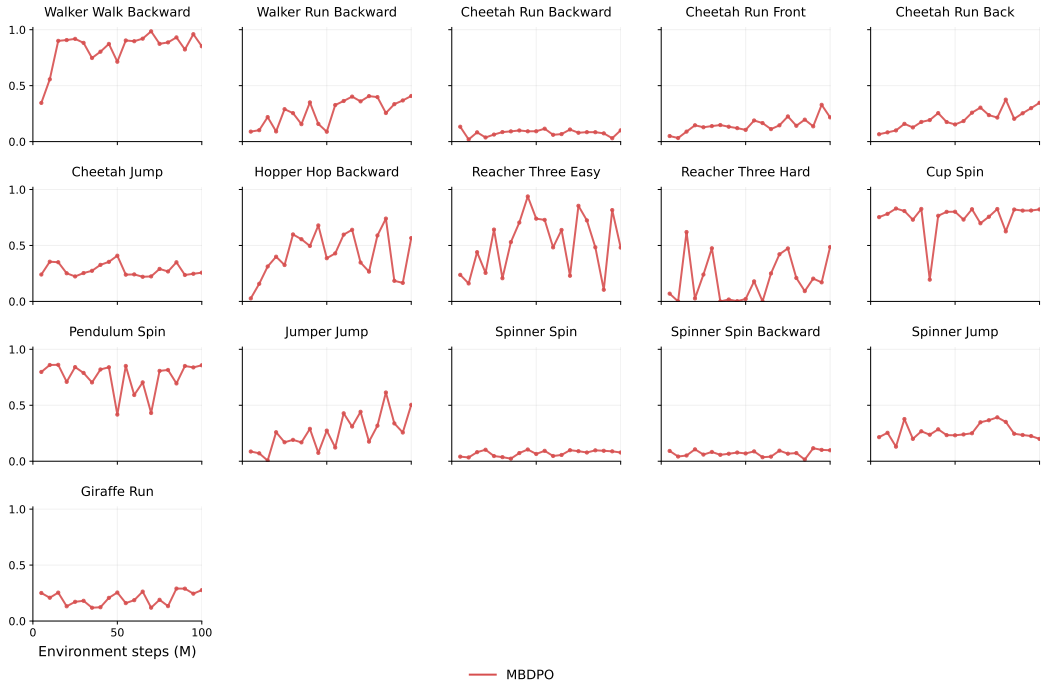

Figure 30: **DMControl Extended.** Per-task normalized scores for 16 tasks.

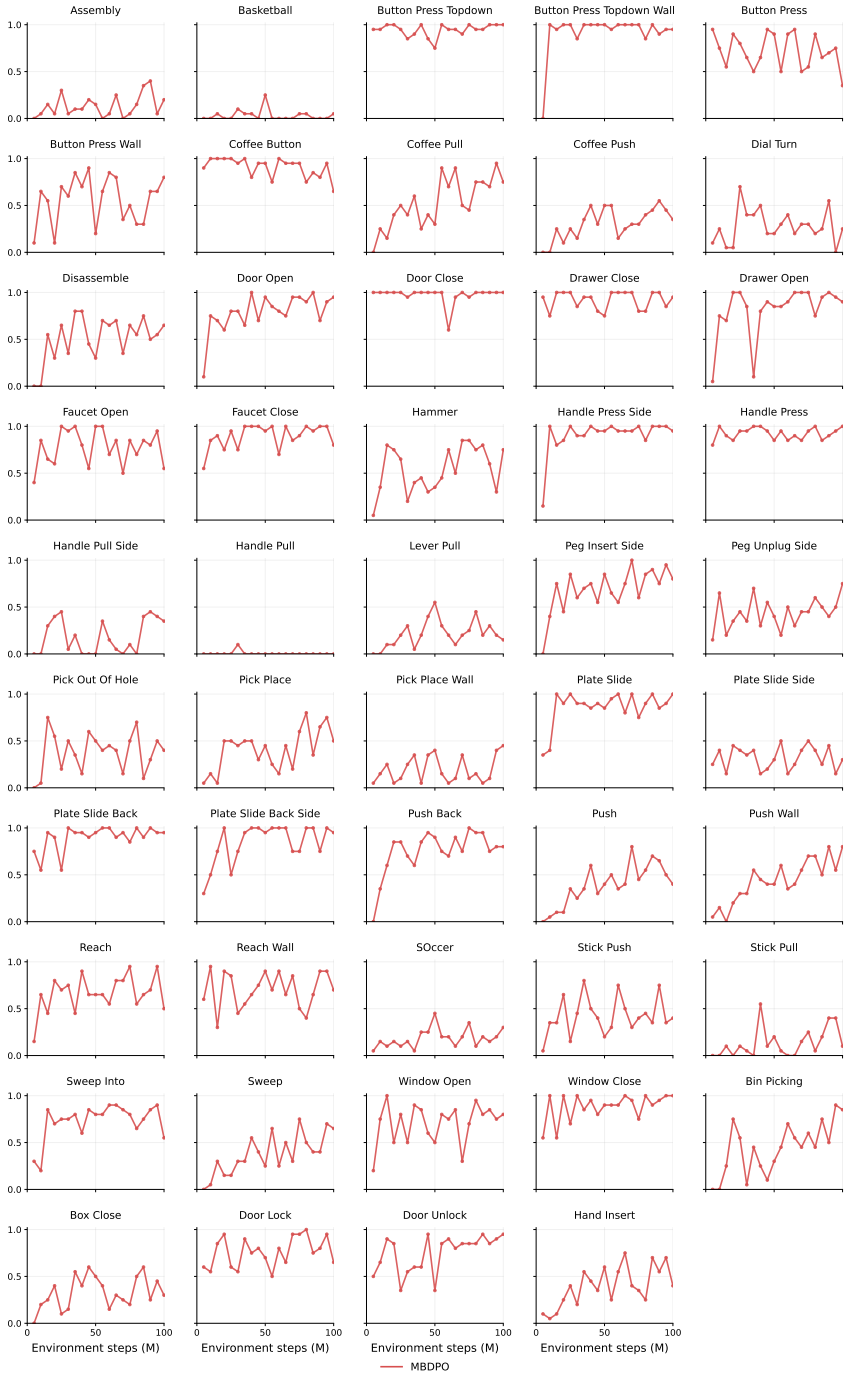

Figure 31: **Meta-World.** Per-task normalized scores for 49 tasks.

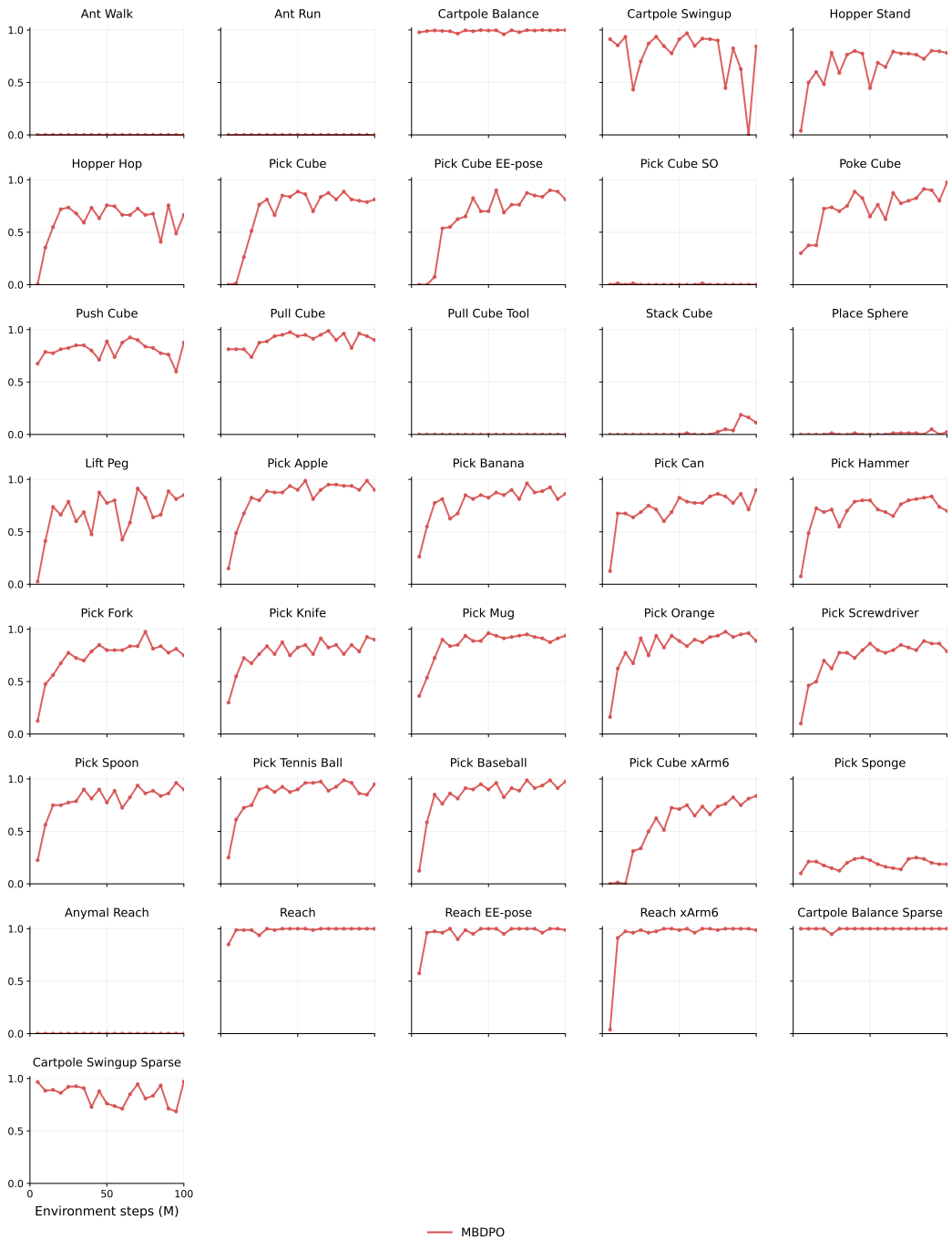

Figure 32: **ManiSkill.** Per-task normalized scores for 36 tasks.

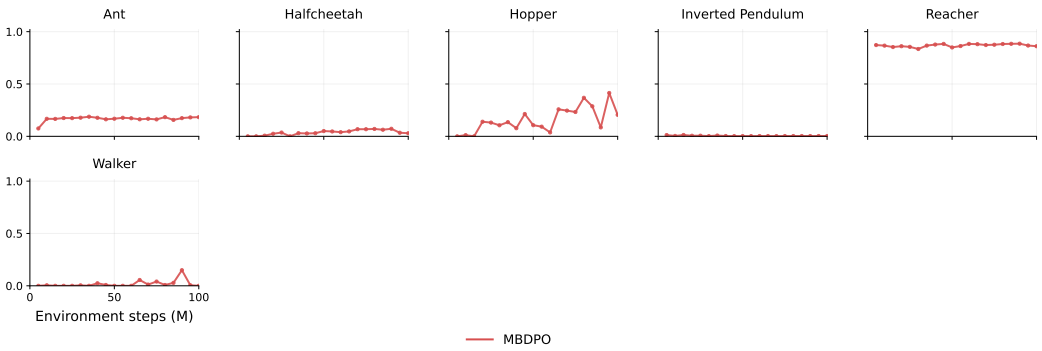

Figure 33: **MuJoCo.** Per-task normalized scores for 6 tasks.

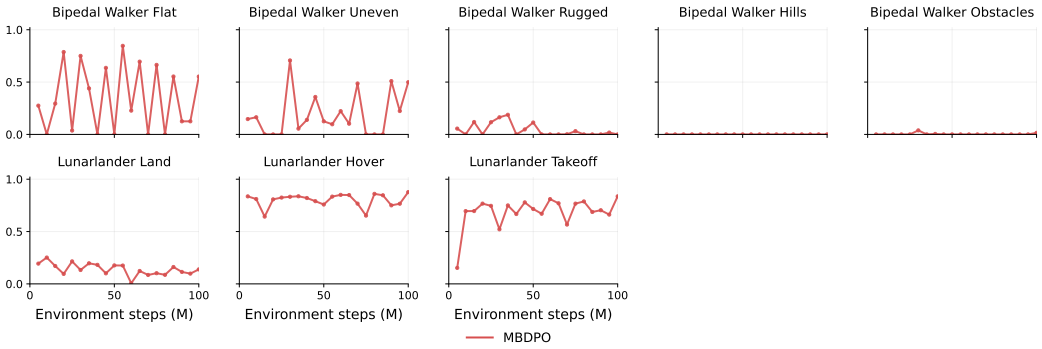

Figure 34: **Box2D.** Per-task normalized scores for 8 tasks.

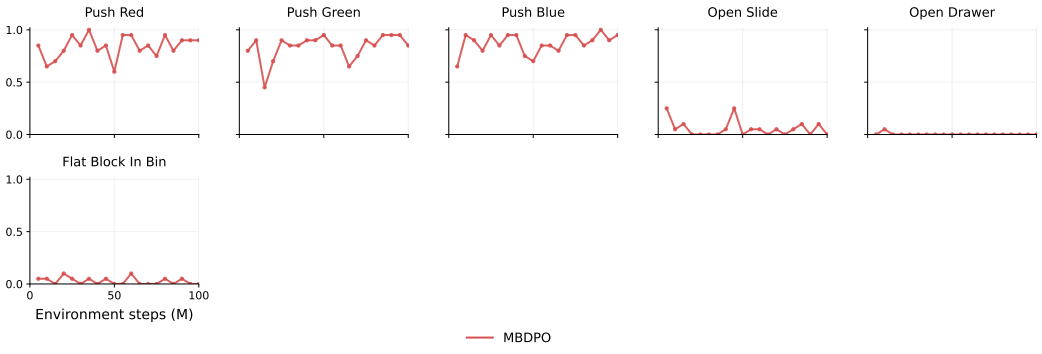

Figure 35: **RoboDesk.** Per-task normalized scores for 6 tasks.

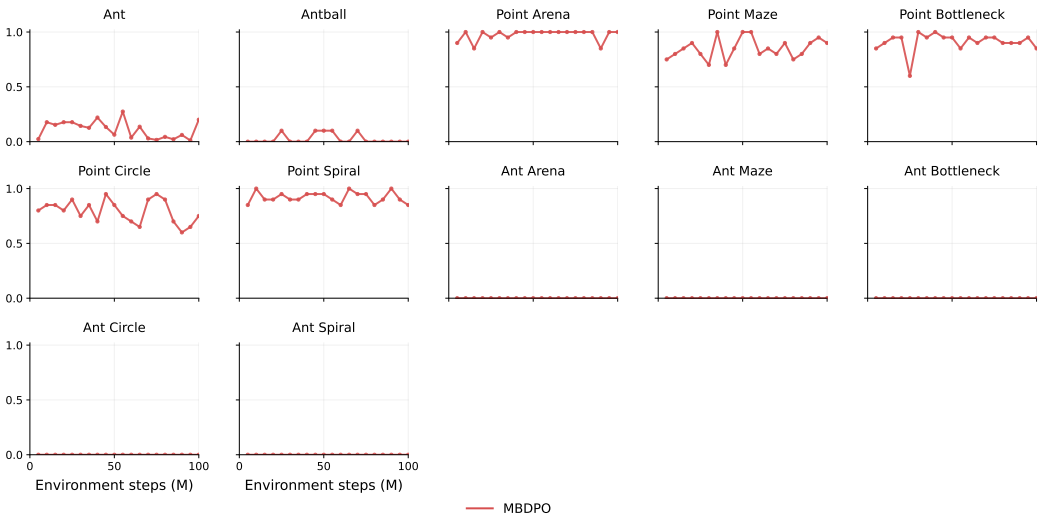

Figure 36: **OGBench.** Per-task normalized scores for 12 tasks.

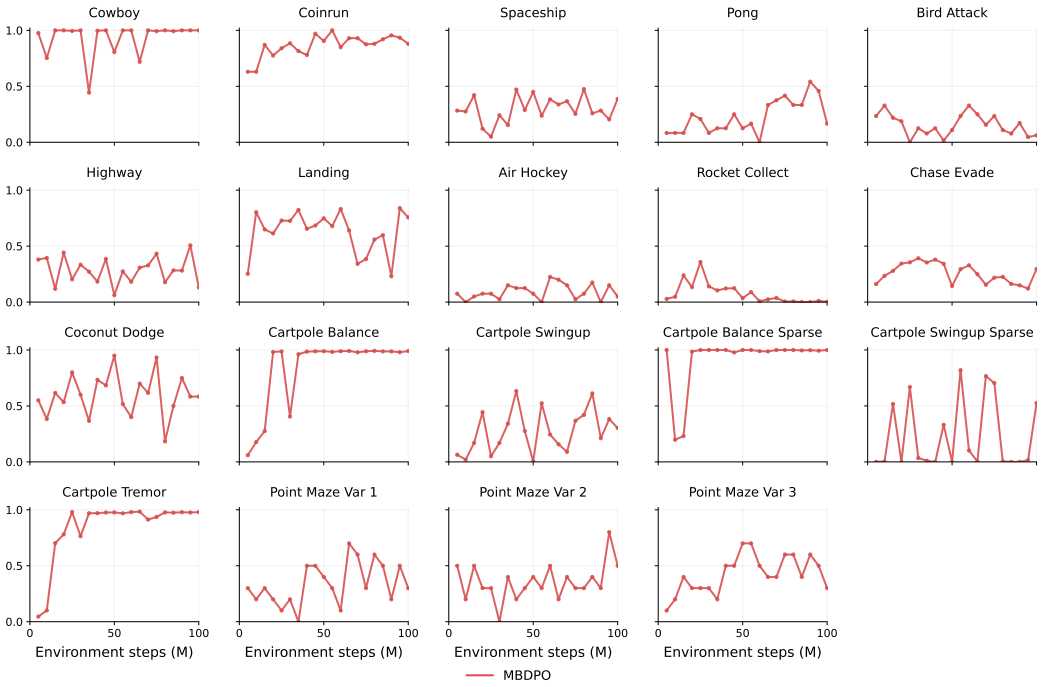

Figure 37: **MiniArcade.** Per-task normalized scores for 19 tasks.



Figure 38: **Atari.** Per-task normalized scores for 27 tasks.

G.3.4 COMPUTATIONAL COST

Table 4 reports the practical wall-clock cost of online training across model sizes. The 8.1M–366.7M models use the same four-A800 configuration and are directly comparable. The 2.4M and 1.82B models use different hardware configurations and are included to document their actual training cost rather than hardware-normalized scaling efficiency.

Table 4: **Wall-clock training time on MMBench.** Time per 1M environment steps is measured during the final continuous online-training stage.

Model size	**2.4M**	**8.1M**	**22.5M**	**88.3M**	**366.7M**	**1.82B**
GPU	A800	A800	A800	A800	A800	H200
Number of GPUs	2	4	4	4	4	8
Time per 1M steps (h)	0.922	1.604	1.948	3.680	9.143	9.281

H Practical Implementation and Hyperparameters

H.1 Complete Algorithm

Algorithm 1: Model-Based Diffusion Policy Optimization (MBDPO)

Input: Encoder $\mathcal{E}$, latent dynamics $\mathcal{F}$, task embedding $\mathcal{E}_{env}$, reward function $\mathcal{R}$, value function $\hat{Q}$, implicit energy function E, score network $\hat{\phi}$, diffusion steps N, Monte Carlo samples T

Initialize: Replay buffer $\mathcal{D} \leftarrow \emptyset$

// Warmup: Train the initial world model
for $i = 1, 2, \ldots, N_{\text{warmup}}$ **do**
  Sample random action $a_h \sim \text{Uniform}(\mathcal{A})$
  Execute action: $(s_h, \square) \leftarrow \text{env.step}(a_h)$
  Store transition (s_h, a_h, r_h, s_{h+1}) into $\mathcal{D}$
Update $\mathcal{E}, \mathcal{F}, \mathcal{R}, \mathcal{E}_{env}, \hat{Q}, E$ by minimizing the model loss in equation 14 using $\mathcal{D}$
for *iteration* $k = 1, 2, \ldots, K$ **do**
  // Step 1: Sampling data from the environment via the score network
  Encode current observation: $z_h \leftarrow \mathcal{E}(s_h)$
  Sample action a_h iteratively via the reverse-time transition kernel in equation 3
  Execute action: $(s_h, \square) \leftarrow \text{env.step}(a_h)$
  Store transition (s_h, a_h, r_h, s_{h+1}) into $\mathcal{D}$
  // Step 2: World model update
  Sample a batch of transitions $\mathcal{B} \sim \mathcal{D}$
  Update $\mathcal{E}, \mathcal{F}, \mathcal{R}, \mathcal{E}_{env}, \hat{Q}, E$ by minimizing the joint loss in equation 14 on batch $\mathcal{B}$
  // Step 3: Diffusion policy optimization
  Estimate the score function via equation 15 using imagined trajectories in the world model
  Update score network $\hat{\phi}$ by minimizing the supervised L^2 loss in equation 16

H.2 Training Regimes

Note that Algorithm 1 outlines the *online* learning paradigm. In the *offline* pretraining setting, the agent does not interact with the environment to collect new data. Instead, the replay buffer $\mathcal{D}$ is initialized with a static, pre-collected dataset, and Step 1 (sampling data via the score network and environment interaction) is entirely omitted. Consequently, the world model and diffusion policy are optimized, relying solely on the offline dataset.

Furthermore, for the ***offline-to-online fine-tuning*** paradigm, the initial warmup phase is completely bypassed. Instead, the algorithm is initialized with the parameters of the world model and the score network that have been pre-trained on the offline dataset. The agent then proceeds directly to the main iterative loop, executing **Step 1**, **Step 2**, and **Step 3** to continuously interact with the environment and fine-tune the score function and world model.

H.3 Architecture and Hyperparameters

H.3.1 Configurations under the TD-MPC2 Protocol

Table 5: **MBDPO hyperparameters for the default 6M model.** We use the same hyperparameters for all online-from-scratch tasks.

Hyperparameter	Value
Search via diffusion sampling	
Horizon (H)	3
Diffusion steps	5-20 steps
Monte Carlo sample number	512
Temperature κ	0.5
Energy regularized factor η	0.1
Replay buffer	
Capacity	1,000,000
Sampling	Uniform
Architecture (6M)	
Encoder dim	256
MLP dim	512
Latent state dim	512
Task embedding dim	96
Task embedding norm	1
Activation	LayerNorm + Mish
$\hat{Q}$-function dropout rate	1%
Ensemble number of $\hat{Q}$-functions	5
Optimization	
Batch size	256 (online) / 1024 (offline)
Policy prior loss norm.	Moving (5%, 95%) percentiles
Optimizer	Adam
Learning rate	3×10^{-4}
Encoder learning rate	1×10^{-4}
Gradient clip norm	20

Table 6: **Model configurations.** We list the specifications for each model configuration (size) of our multi-task offline experiments. *Encoder dim* is the dimensionality of fully connected layers in the encoder $\mathcal{E}$, *MLP dim* is the dimensionality of layers in all other components, *Latent state dim* is the dimensionality of the latent representation z, *# encoder layers* is the number of layers in the encoder $\mathcal{E}$, *# Q-functions* is the number of learned Q-functions, and *Task embedding dim* is the dimensionality of e from Equation (1). TD-targets are always computed by randomly subsampling two $\hat{Q}$-functions, regardless of the number of $\hat{Q}$-functions in the ensemble. We did not experiment with other model configurations. *The default (`base`) configuration used in our single-task RL experiments has 6M parameters.

	1.7M	**6M***	**21M**	**54M**	**340M**
Encoder dim	256	256	1024	1792	4096
MLP dim	384	512	1024	1792	4096
Latent state dim	128	512	768	768	1376
# encoder layers	2	2	3	4	5
# $\hat{Q}$-functions	2	5	5	5	8
Task embedding dim	96	96	96	96	96

H.3.2 CONFIGURATIONS ON MMBENCH

Table 7: **MBDPO hyperparameters for the default 22.5M model on MMBench.** The same configuration is used across all 200 online tasks.

Hyperparameter	Value
Environment and training	
Jointly trained tasks	200
Observation / maximum action dim	128 / 16
Environment steps	100M
Update-to-data ratio	0.075
Diffusion policy optimization	
Horizon (H)	3
Denoising steps (N)	20
Monte Carlo samples (T)	512
Elite samples	64
Policy-prior trajectories	24
Temperature κ	0.5
Energy shaping weight η	0.1
Demonstrations and replay buffer	
Online / demonstration capacity	10M / 1.9M
Demonstration/online sampling ratio	1:1
Demonstration pretraining updates	200K
Random buffer seeding	5 rollouts per environment
Pretraining policy objective	Behavior cloning
Architecture (22.5M)	
Encoder / MLP / latent dim	1024 / 1024 / 512
Encoder / other MLP layers	3 / 2
Task language embedding	512, precomputed and frozen
Task conditioning	Concatenation to model inputs
Activation	LayerNorm + Mish
Latent normalization	SimNorm, group size 8
$\hat{Q}$ ensemble / TD-target subset	5 / 2
Training objectives	
Consistency / reward / value coefficients	20 / 0.1 / 0.1
Policy-prior / energy coefficients	10 / 1
Temporal loss weighting ρ	0.5
Policy entropy coefficient	10^{-4}
Optimization	
Effective batch size	1024
Optimizer	Adam
Learning rate / encoder learning rate	3×10^{-4} / 9×10^{-5}
Warmup updates	5K
Gradient clip norm	20
Target-network update rate τ	0.01
Discount factor	Task-dependent, clipped to $[0.95, 0.995]$

Table 8: **MBDPO model configurations on MMBench.** We report the six model-size configurations used in the online scaling experiments. *Encoder dim* is the hidden width of the state encoder, *MLP dim* is the hidden width shared by the dynamics, reward, contrastive energy, policy-prior, and Q networks, and *Latent state dim* is the dimensionality of z. All non-encoder components contain two hidden layers. The 512-dimensional task language embeddings are precomputed and frozen. TD-targets are computed by randomly subsampling two $\hat{Q}$-functions, regardless of ensemble size. *The 22.5M configuration is the default model.

	2.4M	**8.1M**	**22.5M***	**88.3M**	**366.7M**	**1.82B**
Configuration name	S	B	L	XL	XXL	XXXL
Encoder dim	128	256	1024	2048	4096	8192
MLP dim	256	512	1024	2048	4096	8192
Latent state dim	384	512	512	704	1024	2048
# encoder layers	2	2	3	4	5	6
# hidden layers in other MLPs	2	2	2	2	2	2
# $\hat{Q}$-functions	3	5	5	7	9	13
Task language embedding dim	512	512	512	512	512	512
SimNorm group size	8	8	8	8	8	8
Maximum action dimension	16	16	16	16	16	16
Reward/Q distributional bins	101	101	101	101	101	101