DPMPC: LEARNING DIFFUSION PRIORS FOR MODEL PREDICTIVE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Model predictive control (MPC) has achieved notable success in continuous control. MPC often uses a policy prior to propose candidate trajectories and to learn the value function. When the prior differs from the executed planner, this *actor divergence* biases the value estimates of candidate trajectories. The commonly used Gaussian prior cannot represent the non-Gaussian action distributions induced by stochastic planning, leaving a residual divergence. To address this, we propose DPMPC (Diffusion Priors for Model Predictive Control), which trains an expressive diffusion prior to match the planner's action distribution. Generalized lazy reanalyze estimates this distribution from multiple recent MPC realizations stored per replay state, without rerunning MPC at every policy update. An advantage-weighted score-matching objective then fits the prior to this estimate while emphasizing high-value actions. DPMPC achieves state-of-the-art asymptotic performance and sample efficiency across 14 continuous-control tasks from DMControl, MyoSuite, and HumanoidBench. Further analysis shows that DPMPC yields more accurate value estimates than Gaussian-prior baselines.

1 INTRODUCTION

Reinforcement learning (RL) has achieved impressive results in domains spanning robotics (Zitkovich et al., 2023; Liang et al., 2025), games (Schrittwieser et al., 2020; Hafner et al., 2025), and scientific and engineering applications (Mankowitz et al., 2023; Lagemann et al., 2026). Yet these successes often require large amounts of interaction data, which hinders real-world deployment (Ye & Elsheikh, 2025; Wu et al., 2023). Planning-driven model-based reinforcement learning (MBRL), which combines a learned world model with planning, has emerged as one of the most sample-efficient paradigms (Schrittwieser et al., 2020; Hansen et al., 2022).

Planning-driven MBRL methods typically run model predictive control (MPC) over a short horizon and score each trajectory with a terminal value from a learned value function (Hansen et al., 2022; 2023). Planning performance thus depends directly on the accuracy of this value. The value function is learned with bootstrapped targets whose actions come from a policy prior, the same network that proposes candidate actions for MPC. It therefore estimates the value of the prior rather than that of the executed planner. The mismatch between the prior and the planner's action distribution, known as *actor divergence* (Sikchi et al., 2021; Zhan et al., 2025), directly biases the value estimate. Recent methods reduce this divergence by aligning a Gaussian prior with the planner, either through imitation (Wang et al., 2025) or likelihood maximization (Zhan et al., 2025). Sampling-based MPC, however, solves a non-convex problem: independent runs from the same state can reach different high-return actions, so the planner's action distribution is a non-Gaussian, potentially multimodal mixture (Sasaki & Matsubara, 2021; Zhan et al., 2025). A Gaussian prior cannot represent this mixture, leaving a residual actor divergence that biases the value estimate.

This limitation motivates more expressive policy priors based on diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021), which have proven effective at capturing multimodal action distributions in imitation learning (Chi et al., 2025; Reuss et al., 2023) and offline RL (Wang et al., 2023). These successes, however, depend on a fixed dataset whose actions directly serve as denoising targets. In online RL, no such dataset exists: the target distribution is unknown (Ma et al., 2025). The few online methods therefore rely on a learned critic, either backpropagating value gradients through the denoising chain (Wang et al., 2024) or optimizing critic-guided surrogate

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

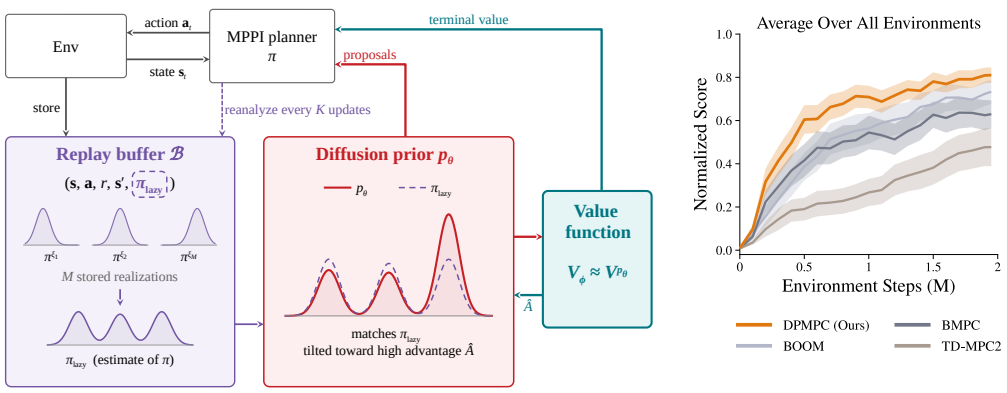


Figure 1: **Left: Overview of DPMPC.** The MPPI planner π acts in the environment, guided by proposals from the diffusion prior p_θ and terminal values from V_ϕ. The prior is trained to match the planner distribution, estimated as a uniform mixture π_{lazy} of M MPC realizations stored in the replay buffer. Because V_ϕ is learned under the prior ($V_\phi \approx V^{p_\theta}$), matching p_θ to π makes it an accurate terminal value for the planner. **Right: Average performance** over 14 tasks of DPMPC and baselines.

objectives (Psenka et al., 2024; Ding et al., 2024; Ma et al., 2025; Dong et al., 2025). Because the target is specified only implicitly through a nonstationary value estimate, training can be costly, unstable, or sample-inefficient. Moreover, these methods are predominantly model-free, and diffusion policy priors for planning-driven MBRL remain largely unexplored.

Our key insight is that, in planning-driven MBRL, the planner itself can supply an explicit training target: its action distribution, which is exactly what the prior must match to reduce *actor divergence*. Matching this distribution turns online diffusion policy training into a supervised learning problem. Building on this insight, we propose **D**iffusion **P**riors for **M**odel **P**redictive **C**ontrol (**DPMPC**). A naive implementation, however, would rerun MPC to sample the planner distribution at every policy update, which is computationally prohibitive. We address this with generalized lazy reanalyze, which stores multiple MPC realizations per replay state and periodically refreshes them to estimate the planner distribution. We then train the diffusion policy prior on samples from this estimate with an advantage-weighted score-matching loss that shifts mass toward higher-value actions while keeping the prior close to the planner distribution.

In this paper, we make three key contributions: (1) We identify the residual *actor divergence* left by Gaussian policy priors as a source of value bias in planning-driven MBRL. We show that the value estimates of recent Gaussian-prior methods retain a large error relative to the planner's return throughout training. (2) We develop a simple yet effective recipe for training diffusion policy priors. It combines generalized lazy reanalyze, which maintains an estimate of the planner distribution, with an advantage-weighted score-matching loss that trains the prior toward high-value actions while keeping it close to this distribution. (3) We demonstrate that DPMPC achieves state-of-the-art performance across 14 tasks from DMControl (Tassa et al., 2018), MyoSuite (Caggiano et al., 2022), and HumanoidBench (Sferrazza et al., 2024). Our analysis shows that DPMPC provides more accurate value estimates than Gaussian-prior baselines.

2 Background

We consider a Markov Decision Process (MDP) (Puterman, 2014) defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1]$ is the discount factor, and $\mu \in \Delta(\mathcal{S})$ is the initial state distribution. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ defines a distribution over actions given states. The objective is to find an optimal policy that maximizes the expected discounted return $\mathbb{E}_{s_0 \sim \mu,\, a_t \sim \pi(\cdot|s_t),\, s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)} [\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)]$.

2.1 PLANNING-DRIVEN MBRL

Planning-driven MBRL methods (Hansen et al., 2023; Wang et al., 2025; Zhan et al., 2025) combine a latent-space world model with MPC. The agent does not execute the parametric policy directly; instead, it uses the policy as a prior $p(\mathbf{a} \mid \mathbf{z})$ to guide online planning. An observation $\mathbf{s}$ is encoded into a latent state $\mathbf{z} = h(\mathbf{s})$, on which the learned dynamics $\mathbf{z}' = d(\mathbf{z}, \mathbf{a})$ and reward $\hat{r} = R(\mathbf{z}, \mathbf{a})$ are defined. We refer to TD-MPC2 (Hansen et al., 2023) for world-model training details.

MPC with a policy prior. The planner is commonly instantiated with Model Predictive Path Integral (MPPI) (Williams et al., 2015). Given the current latent state $\mathbf{z}_t$, MPPI optimizes a Gaussian search distribution $\mathcal{N}(\boldsymbol{\mu}_{t:t+H}, \boldsymbol{\sigma}^2_{t:t+H})$ over action sequences $\mathbf{a}_{t:t+H}$, where H is the planning horizon. The policy prior $p(\mathbf{a} \mid \mathbf{z})$ contributes additional proposal trajectories to the candidate set. All candidates are rolled out through the latent dynamics and evaluated under the finite-horizon objective:

$$(\boldsymbol{\mu}^*_{t:t+H}, \boldsymbol{\sigma}^*_{t:t+H}) = \arg\max_{\boldsymbol{\mu}, \boldsymbol{\sigma}} \mathbb{E}_{\mathbf{a}_{t:t+H} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)} \left[\sum_{h=t}^{t+H-1} \gamma^{h-t} R(\mathbf{z}_h, \mathbf{a}_h) + \gamma^H U(\mathbf{z}_{t+H}, \mathbf{a}_{t+H}) \right], \tag{1}$$

where U is a terminal value estimate from the learned value model. The planner outputs the optimized distribution $\mathcal{N}(\boldsymbol{\mu}^*_{t:t+H}, (\boldsymbol{\sigma}^*_{t:t+H})^2)$, and only its first action is executed in the environment. During training, this action is sampled from the optimized Gaussian distribution to explore the environment, whereas during evaluation the optimized mean $\boldsymbol{\mu}^*_t$ is executed.

Value functions. In the planning objective above, the terminal bootstrap term can be instantiated either as an action-value function $Q(\mathbf{z}_{t+H}, \mathbf{a}_{t+H})$ or as a state-value function $V(\mathbf{z}_{t+H})$. Here, $Q^p(\mathbf{z}_t, \mathbf{a}_t) = \mathbb{E}_p[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid \mathbf{z}_t, \mathbf{a}_t]$ is the expected return of taking action $\mathbf{a}_t$ at latent state $\mathbf{z}_t$ and following the policy prior p thereafter; $V^p(\mathbf{z}_t) = \mathbb{E}_{\mathbf{a}_t \sim p(\cdot|\mathbf{z}_t)}[Q^p(\mathbf{z}_t, \mathbf{a}_t)]$ is the return under the policy prior. The advantage $A^p(\mathbf{z}_t, \mathbf{a}_t) = Q^p(\mathbf{z}_t, \mathbf{a}_t) - V^p(\mathbf{z}_t)$ measures how much an action improves upon the return of the prior. When only a state-value function is modeled explicitly, it can be trained via model-based TD learning (Wang et al., 2025) by constructing an N-step bootstrapped target:

$$\mathcal{L}_V(\phi) = \mathbb{E}_{(s,a,r,s')} \left[\mathrm{CE}\left(V_\phi(\mathbf{z}_t), \hat{V}(\mathbf{z}_t) \right) \right], \quad \hat{V}(\mathbf{z}_t) = \gamma^N V(\mathbf{z}_{t+N}) + \sum_{k=0}^{N-1} \gamma^k R(\mathbf{z}_{t+k}, \mathbf{a}_{t+k}), \tag{2}$$

where $\mathbf{z}_{t+1} = d(\mathbf{z}_t, \mathbf{a}_t)$ evolves through the learned model, $\hat{V}$ is the TD target computed from world-model rollouts, and CE denotes cross-entropy with a discrete regression formulation of the value function.

2.2 DIFFUSION POLICIES

A diffusion policy (Ding et al., 2024; Wang et al., 2024) models a conditional action distribution $p_\theta(\mathbf{a} \mid \mathbf{z})$ by transforming Gaussian noise into an action through an iterative denoising process conditioned on the latent state $\mathbf{z}$ (Sohl-Dickstein et al., 2015; Ho et al., 2020). Given a clean action $\mathbf{a}_0$ and a diffusion step $\tau \in \{1, \ldots, T\}$, a noisy action is constructed as $\mathbf{a}_\tau = \sqrt{\bar{\alpha}_\tau}\, \mathbf{a}_0 + \sqrt{1 - \bar{\alpha}_\tau}\, \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\bar{\alpha}_\tau = \prod_{l=1}^{\tau}(1 - \beta_l)$ for a predefined variance schedule $\beta_1, \ldots, \beta_T$. A denoising network $\boldsymbol{\epsilon}_\theta(\mathbf{a}_\tau, \tau, \mathbf{z})$ is trained to predict the injected noise by minimizing the denoising score-matching objective:

$$\mathbb{E}_{\mathbf{z}, \mathbf{a}_0,\, \tau \sim \mathrm{Unif}\{1,\ldots,T\},\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{a}_\tau, \tau, \mathbf{z}) \right\|^2 \right]. \tag{3}$$

At inference time, an action is sampled by iteratively denoising for T steps conditioned on $\mathbf{z}$.

3 METHODOLOGY

3.1 VALUE ESTIMATION ERROR FROM ACTOR DIVERGENCE

The planner relies on the value function to evaluate candidate trajectories: the terminal term U in Eq. 1 accounts for the return beyond the planning horizon. This value function, however, is learned under

the policy prior, i.e., it estimates V^p, whereas the agent executes the planner. It therefore predicts the planner's return accurately only when the prior and the planner have similar action distributions. Any *actor divergence* between the two biases the value estimate. To analyze this divergence, we first characterize the action distribution of the planner. At latent state $\mathbf{z}$, a single MPC call guided by the prior p returns an optimized Gaussian distribution over action sequences. Its first-action marginal, $\pi^\xi(\cdot \mid \mathbf{z}, p) = \mathcal{N}\big(\boldsymbol{\mu}_0^{*,\xi}(\mathbf{z}, p),\ (\boldsymbol{\sigma}_0^{*,\xi}(\mathbf{z}, p))^2\big)$, depends on the internal randomness ξ of the stochastic MPPI planner. Marginalizing over ξ yields the action distribution of the executed planner:

$$\pi(\cdot \mid \mathbf{z}, p) = \mathbb{E}_\xi\big[\pi^\xi(\cdot \mid \mathbf{z}, p)\big]. \tag{4}$$

Because MPPI optimizes a non-convex objective, independent draws of ξ can converge to different local optima. The planner distribution π is therefore a mixture of Gaussians with distinct means, which is generally non-Gaussian. To see how this affects the value estimate, we decompose its error with respect to the planner's return $V^\pi(\mathbf{z})$, i.e., the expected discounted return when executing π from $\mathbf{z}$:

$$V_\phi(\mathbf{z}) - V^\pi(\mathbf{z}) = \underbrace{V_\phi(\mathbf{z}) - V^p(\mathbf{z})}_{\text{estimation error}} + \underbrace{V^p(\mathbf{z}) - V^\pi(\mathbf{z})}_{\text{actor-divergence bias}}. \tag{5}$$

The first term is the error of V_ϕ in fitting its own target V^p, which is governed by model accuracy and function approximation. The second term arises because the value is learned under the prior rather than the planner, and it vanishes when $p = \pi$. An accurate value estimate thus requires the prior to match the planner distribution, i.e., $p \approx \pi$.

Recent methods reduce the actor divergence by aligning the prior with the planner. BMPC (Wang et al., 2025) minimizes the KL divergence from the planner distribution to the prior, while BOOM (Zhan et al., 2025) maximizes the likelihood of planner actions under the prior. Both methods, however, retain a Gaussian prior, which cannot represent the mixture in Eq. 4. The actor divergence thus persists no matter how well the Gaussian prior is trained.

Table 1: **Value estimation error on H1hand-Run.** We evaluate the learned value network at three checkpoints during training, reporting the relative mean absolute error (%; lower is better), averaged over three seeds.

Method	Environment steps (M)		
	0.4	1.2	2.0
BMPC	82.4	54.6	75.9
BOOM	69.3	59.0	29.4

Table 1 reports the value error of BMPC and BOOM on H1hand-Run. We measure the true return with a Monte Carlo estimate, following the protocol detailed in Appendix B.3. Both Gaussian-prior methods retain a large value error throughout training. The relative error of BMPC stays between 55% and 82%, while that of BOOM decreases from 69% at 0.4M steps to 29% at 2M steps. Although the two methods train their priors and value functions differently, the error persists under both alignment objectives. Removing this error therefore requires a policy prior expressive enough to represent the planner distribution.

3.2 Diffusion Priors for MPC

We propose **DPMPC** (Diffusion Priors for Model Predictive Control), a framework that trains a diffusion policy prior for guiding model-based planning. DPMPC is built on the TD-MPC2 (Hansen et al., 2023) framework, inheriting the latent world model and MPPI planner. We parameterize the policy $p_\theta(\mathbf{a} \mid \mathbf{z})$ as a diffusion model and use stochastic MPC to bootstrap the policy training. DPMPC estimates the planner distribution with generalized lazy reanalyze, trains the diffusion policy prior on this estimate with advantage-weighted score matching, and learns the value function under the diffusion prior. An overview of the DPMPC training loop is shown in Figure 1 and the complete training procedure is summarized in Algorithm 1.

Generalized lazy reanalyze. The training target for the diffusion policy prior is the planner distribution $\pi(\cdot \mid \mathbf{z}, p_\theta)$ in Eq. 4. Sampling directly from this distribution at every gradient step would require repeated MPC calls, incurring substantial planning cost. We therefore store MPC realizations in the replay buffer for reuse across policy updates. Since different stochastic MPC runs can reach different high-value action regions, we allow each replay state to retain up to M realizations. Concretely, for each replay state $\mathbf{s}$ with latent $\mathbf{z}$, the corresponding buffer slot maintains a first-in-first-out (FIFO) queue of $M_\mathbf{z} \le M$ recent MPC realizations $\big\{\pi^{\xi_i}(\cdot \mid \mathbf{z}, p_{\theta_i})\big\}_{i=1}^{M_\mathbf{z}}$. During

Algorithm 1 DPMPC Training

Require: p_θ, h, d, R, V_ϕ: randomly initialized networks
$\mathcal{B}$, K, M: replay buffer, reanalyze interval, queue capacity

while not converged **do**
  // Collect experience:
  for step $t = 0 \dots T_{\text{env}}$ **do**
    $\pi_t^\xi \leftarrow \pi^\xi(\cdot \mid h(\mathbf{s}_t),\, p_\theta)$; $\pi_{\text{lazy},t} \leftarrow \{\pi_t^\xi\}$ ◁ *Run MPC once; initialize queue*
    $\mathbf{a}_t \sim \pi_t^\xi$ ◁ *Sample and execute action*
    $(\mathbf{s}_{t+1},\, r_t) \leftarrow \text{env.step}(\mathbf{a}_t)$ ◁ *Step environment*
    $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{(\mathbf{s}_t,\, \mathbf{a}_t,\, r_t,\, \mathbf{s}_{t+1},\, \pi_{\text{lazy},t})\right\}$ ◁ *Add to buffer*
  // Update networks:
  for num updates per episode **do**
    $\left(\mathbf{s}_t,\, \mathbf{a}_t,\, r_t,\, \mathbf{s}_{t+1},\, \pi_{\text{lazy},t}\right)_{t:t+H} \sim \mathcal{B}$ ◁ *Sample trajectory*
    Update h, d, R; update V_ϕ via Eq. 2 ◁ *World model and value update*
    Update p_θ via Eq. 7 ◁ *Diffusion policy prior update*
    // Generalized lazy reanalyze:
    if update_step $\bmod K = 0$ **then**
      $\pi^\xi_{t:t+H} \leftarrow \pi^\xi(\cdot \mid h(\mathbf{s}_{t:t+H}),\, p_\theta)$ ◁ *Re-run MPC once per state*
      $\mathcal{B} \overset{\text{enqueue}}{\longleftarrow} \pi^\xi_{t:t+H}$ ◁ *Append to FIFO queues; evict oldest if full*

environment interaction, the MPC call at $\mathbf{z}_t$ initializes the queue with a single realization ($M_{\mathbf{z}} = 1$). At fixed update intervals, we rerun MPC once for each state in a sampled batch using fresh planner randomness, append the new realization to its queue, and evict the oldest realization if the queue is full. The stored realizations define the empirical estimate of the planner distribution through their uniform mixture: $\pi_{\text{lazy}}(\cdot \mid \mathbf{z}) := \frac{1}{M_{\mathbf{z}}} \sum_{i=1}^{M_{\mathbf{z}}} \pi^{\xi_i}(\cdot \mid \mathbf{z}, p_{\theta_i})$. This procedure extends the *lazy reanalyze* mechanism of BMPC to support multiple MPC realizations per state.

Advantage-weighted score matching. Some MPC realizations can be stale since the buffer stores multiple realizations per state and the refresh rate is limited. In policy training, we therefore seek a training objective that emphasizes actions with higher estimated value while keeping the prior close to the planner distribution, so that the value learned under the prior remains accurate for the planner. By the definition of advantage in Section 2.1, emphasizing higher-value actions is equivalent to emphasizing higher-advantage actions. Let $d_{\mathcal{B}}(\mathbf{z})$ denote the latent-state distribution induced by $\mathcal{B}$. We formulate policy training as a constrained optimization problem:

$$\begin{aligned} \arg\max_{p_\theta} \quad & \int_{\mathbf{z}} d_{\mathcal{B}}(\mathbf{z}) \int_{\mathbf{a}} p_\theta(\mathbf{a}|\mathbf{z})\, A(\mathbf{z}, \mathbf{a})\, d\mathbf{a}\, d\mathbf{z} \\ \text{s.t.} \quad & D_{\text{KL}}\Big(p_\theta(\cdot|\mathbf{z}) \,\Big\|\, \pi_{\text{lazy}}(\cdot|\mathbf{z})\Big) \le \epsilon, \quad \forall\, \mathbf{z}. \end{aligned} \tag{6}$$

Solving this problem in closed form and projecting the solution onto the diffusion policy prior yields an advantage-weighted score-matching objective. The full derivation is deferred to Appendix A. We train the diffusion policy prior with this objective on H-step latent rollouts of the learned world model and apply temporal weighting λ^t along the rollout horizon, which yields the final DPMPC loss

$$\begin{aligned} \mathcal{L}_{\text{DPMPC}}(\theta) = \mathbb{E}_{(\mathbf{s}, \mathbf{a}, \pi_{\text{lazy}})_{0:H} \sim \mathcal{B}} \Bigg[\sum_{t=0}^{H} \lambda^t \\ \cdot\ \mathbb{E}_{\substack{\tau \sim \text{Unif}\{1,\dots,T\},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{a}_t^{\text{lazy}} \sim \pi_{\text{lazy},t}(\cdot|\mathbf{z}_t)}} \left[w(\mathbf{z}_t, \mathbf{a}_t^{\text{lazy}}) \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\Big(\sqrt{\bar{\alpha}_\tau}\, \mathbf{a}_t^{\text{lazy}} + \sqrt{1 - \bar{\alpha}_\tau}\, \boldsymbol{\epsilon},\ \tau,\ \mathbf{z}_t\Big) \right\|^2 \right] \Bigg], \\ w(\mathbf{z}_t, \mathbf{a}_t^{\text{lazy}}) = \exp\left(\frac{\hat{A}(\mathbf{z}_t, \mathbf{a}_t^{\text{lazy}})}{\eta\, S} \right), \end{aligned} \tag{7}$$

where $\mathbf{z}_0 = h(\mathbf{s}_0)$ and $\mathbf{z}_{t+1} = d(\mathbf{z}_t, \mathbf{a}_t)$ are rolled out on the world model. Here, $w(\mathbf{z}_t, \mathbf{a}_t^{\text{lazy}})$ is the per-sample advantage weight and $\eta > 0$ is the advantage temperature. Small η sharpens the

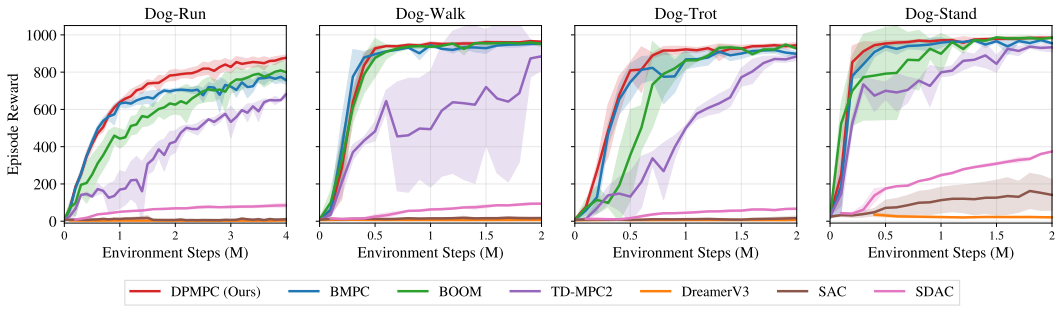


Figure 2: **Training curves on DMControl.** Dog-Run is trained for 4M environment steps due to its slower convergence, while all other tasks are trained for 2M steps.

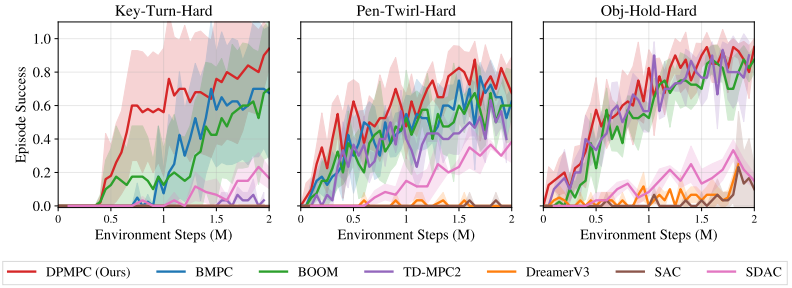


Figure 3: **Training curves on MyoSuite.** Performance on the dexterous manipulation tasks is reported as success rate.

weighting toward high-advantage actions; in the limit $\eta \to +\infty$, all weights collapse to a constant and the objective reduces to imitation of π_{lazy}. We normalize the advantage by a running scale S, a moving estimate of the spread of $\hat{A}$ over training batches, so that a single temperature η transfers across tasks with different reward scales. The weights are also clamped to a finite range for training stability. The advantage is estimated from the learned world model via the one-step bootstrap $\hat{A}(\mathbf{z}, \mathbf{a}) = R(\mathbf{z}, \mathbf{a}) + \gamma V_\phi(d(\mathbf{z}, \mathbf{a})) - V_\phi(\mathbf{z})$, where V_ϕ is the learned latent state-value function.

Value function training. DPMPC trains V_ϕ with the model-based TD objective in Eq. 2, sampling every rollout action from the diffusion policy prior. The value function V_ϕ therefore approximates V^{p_θ}. Since the diffusion prior is trained to match the planner distribution, V^{p_θ} provides an estimate of the value of the planner.

4 Experiments

4.1 Experimental Setup

We compare DPMPC with six representative online RL baselines, including two model-free methods and four model-based methods. For the model-free baselines, we choose **SAC** (Haarnoja et al., 2018), which is built on a Gaussian policy under the maximum-entropy RL framework, and **SDAC** (Ma et al., 2025), a diffusion-policy-based RL method. For the model-based methods, **DreamerV3** (Hafner et al., 2025) represents imagination-based MBRL, while **TD-MPC2** (Hansen et al., 2023), **BMPC** (Wang et al., 2025), and **BOOM** (Zhan et al., 2025) are planning-driven approaches that differ in how they train the Gaussian policy prior. Experiments are conducted on 14 challenging high-dimensional tasks collected from DMControl (Tassa et al., 2018), MyoSuite (Caggiano et al., 2022), and Humanoid-Bench (Sferrazza et al., 2024). Complete environment specifications and hyperparameter settings are provided in Appendix B.

Table 2: **Training time.** Wall-clock hours for 2M environment steps on a single RTX 3090 GPU.

Task	TD-MPC2	BMPC	BOOM	DPMPC (ours)
Dog-Run	5.3	7.5	20.1	9.6
H1hand-Run	11.0	15.6	40.4	19.3

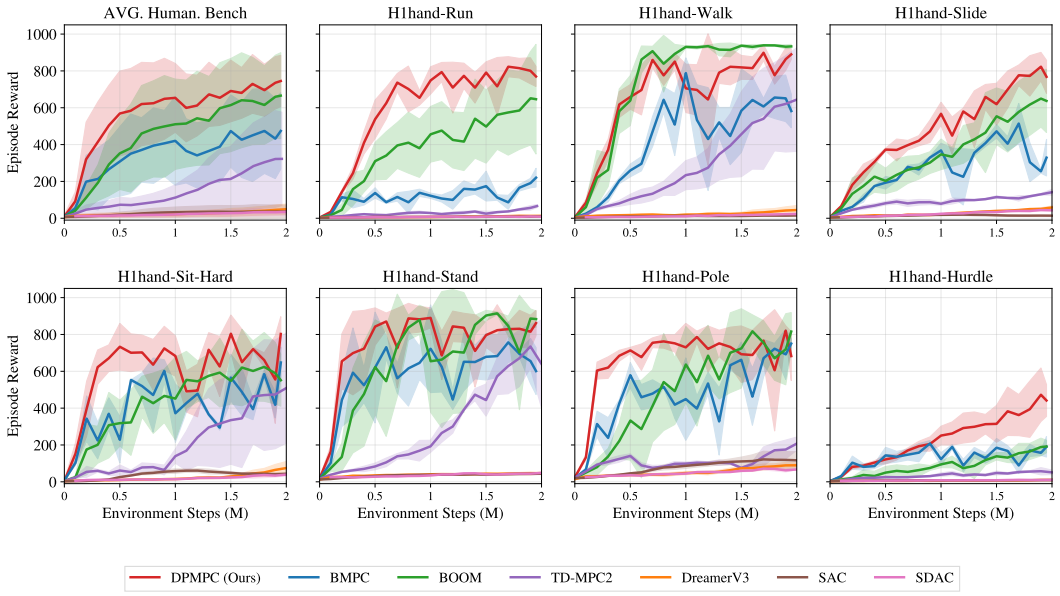


Figure 4: **Training curves on HumanoidBench.** The H1hand tasks form a challenging high-dimensional control setting with a 61-dimensional action space. The top-left panel reports the average episode return across all seven tasks.

4.2 Comparative Evaluation

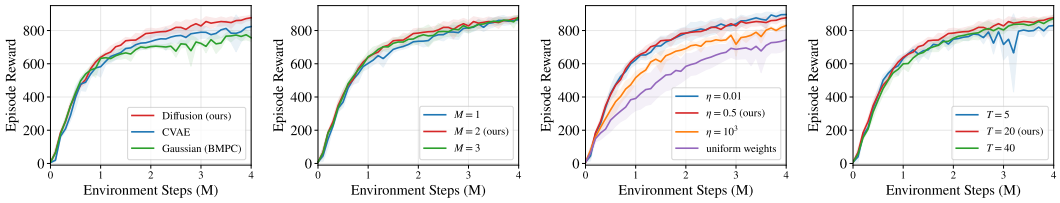
We report training curves across three benchmarks in Figures 2–4, averaged over four seeds. DPMPC consistently outperforms the model-free baselines (SAC, SDAC) and the imagination-based DreamerV3 across all benchmarks. Among the planning-driven baselines, DPMPC achieves the highest asymptotic return and sample efficiency on DMControl and MyoSuite. On the 61-dimensional H1hand tasks, DPMPC outperforms every baseline with an average return of **745**. This marks a **+12.0%** improvement over BOOM, the strongest competing planner, and a **+57.1%** gain over BMPC. The margin widens to **+130.9%** over TD-MPC2, suggesting that the diffusion policy prior is especially effective for planning in high-dimensional action spaces. These gains come at a moderate computational cost: DPMPC requires about $1.3\times$ the wall-clock training time of BMPC and less than half that of BOOM (Table 2; see Appendix E for the hardware setup).

4.3 Value Estimation Analysis

Table 3 repeats the diagnostic of Section 3.1 for all three methods on Dog-Run and H1hand-Run. On H1hand-Run, DPMPC attains the lowest value error at every checkpoint. Its relative error drops from 51% at 0.4M steps to 6.5% at 1.2M steps and stays below 7% thereafter. BMPC and BOOM retain errors of at least 29% throughout training. On Dog-Run, BMPC is more accurate at 0.8M steps, but DPMPC reduces its error to 2% at 2.4M steps and 1% at 4M steps, whereas BMPC and BOOM plateau at about 5% and 10%, respectively. The advantage of DPMPC in value accuracy is thus largest on the task with the higher-dimensional action space (61 dimensions in H1hand-Run versus 38 in Dog-Run), which is also where DPMPC achieves its largest performance gains. We attribute the remaining error of DPMPC on both tasks partly to generalized lazy reanalyze. Because the planner keeps evolving during training, the stored MPC realizations lag behind the current planner,

Table 3: **Value estimation analysis.** Relative mean absolute error (%) of the learned value at three training checkpoints (lower is better). Entries report the mean ± standard deviation over three seeds. Bold marks the lowest mean error in each column.

	Dog-Run			H1hand-Run		
Method	0.8M	2.4M	4M	0.4M	1.2M	2M
BMPC	**7.1**±5.7	5.5±0.7	5.4±3.7	82.4±34.1	54.6±21.4	75.9±35.3
BOOM	12.0±5.5	11.1±3.4	9.5±1.7	69.3±3.3	59.0±13.9	29.4±1.8
DPMPC (ours)	12.4±2.7	**1.8**±1.1	**1.0**±0.6	**50.6**±6.1	**6.5**±2.6	**6.8**±5.6



(a) (b) (c) (d)

Figure 5: **Ablation studies on Dog-Run.** (a) Policy prior parameterization. (b) Number of MPC realizations stored per state M. (c) Advantage temperature η. (d) Number of diffusion steps T.

so π_{lazy} tracks the planner distribution only approximately. Appendix D further shows that the learned diffusion prior produces non-Gaussian, often multimodal action distributions.

4.4 Ablation Studies

Effect of the policy parameterization. Figure 5a compares three parameterizations of the policy prior: the diffusion policy of DPMPC, a conditional variational autoencoder (CVAE) (Kingma & Welling, 2019), and a Gaussian policy. The CVAE replaces the diffusion policy within DPMPC, leaving all other components unchanged. The Gaussian policy follows the BMPC training protocol. The CVAE outperforms the Gaussian policy, and the diffusion policy achieves the highest return. This result shows that the expressiveness of the policy prior is important for control performance.

Effect of the number of MPC realizations. Figure 5b varies the number of MPC realizations $M \in \{1, 2, 3\}$ stored per buffer state during generalized lazy reanalyze. Setting $M = 2$ yields the best sample efficiency and shows stable training. With $M = 1$, each target reflects only a single MPC realization. This choice slows learning, likely because the single realization fails to capture the planner distribution. Increasing M to 3 brings no further gain, possibly because older realizations become stale. Appendix C further isolates the periodic refresh of generalized lazy reanalyze.

Effect of advantage temperature. Figure 5c examines the advantage temperature η alongside an unweighted variant that assigns uniform weights to all targets. Removing the weighting is the most damaging setting: it learns most slowly throughout training and attains the lowest asymptotic return. With $\eta = 10^3$ the weights become nearly uniform and performance degrades toward that variant, confirming that the gains stem from advantage weighting rather than from score matching on π_{lazy} alone. Sharpening the weighting from $\eta = 0.5$ to $\eta = 0.01$ leaves performance essentially unchanged, indicating that DPMPC is robust to η over this range.

Effect of the number of diffusion steps. Figure 5d varies the number of denoising steps of the diffusion policy prior. With $T = 5$, the short denoising chain limits the expressiveness of the prior, which leads to unstable learning after 2.5M steps and a lower final return. Increasing T to 40 slightly slows learning in the intermediate phase without improving the final return, while doubling the cost of sampling from the prior.

5 Related Work

Model-based Reinforcement Learning. Model-based reinforcement learning improves sample efficiency by learning a dynamics model and using it to support decision-making. Imagination-based methods, such as Dreamer and DIAMOND (Hafner et al., 2025; Alonso et al., 2024), optimize policies with synthetic rollouts generated by the learned model and have achieved strong results in visual control. Planning-driven methods such as the TD-MPC family, BMPC, and BOOM combine learned latent models with MPC and use a learned policy prior to guide trajectory optimization (Hansen et al., 2022; 2023; Wang et al., 2025; Zhan et al., 2025). While these methods achieve strong sample efficiency, they learn the value function under the policy prior but execute the planner, and this *actor divergence* (Sikchi et al., 2021; Zhan et al., 2025) biases the value estimate. Their policy priors are typically Gaussian, which cannot represent the non-Gaussian action distribution of the planner and thus leaves a residual actor divergence. The closest prior work is BMPC, which narrows this divergence by imitating the planner distribution with a Gaussian prior, yet a residual divergence remains. Our approach, DPMPC, addresses this limitation by using a diffusion policy prior that can capture the multimodal planner distribution and reduce the actor divergence.

Diffusion Policies for Online RL. Diffusion policies have recently been adopted as expressive actors for online reinforcement learning, where their iterative denoising process can represent complex action distributions beyond standard Gaussian policies (Wang et al., 2024; Ding et al., 2024; Psenka et al., 2024; Ma et al., 2025; Dong et al., 2025). Existing methods are largely model-free and couple diffusion policy optimization to a learned critic. One line of work maximizes value estimates by backpropagating through the denoising chain (Wang et al., 2024), but this direct approach is computationally expensive and can be unstable because gradients must pass through many sequential sampling steps. Another line avoids full backpropagation by using critic-guided objectives, including Q-score matching (Psenka et al., 2024) and Q-weighted variational objectives (Ding et al., 2024). Although effective, these methods define target action distributions only implicitly through the learned value function, so policy updates are typically formed from samples drawn from the current policy rather than from explicit high-value action targets, which can make diffusion policy learning less direct and less sample efficient (Ma et al., 2025). In contrast, DPMPC uses model-based planning to produce explicit high-value action targets for training the diffusion policy prior.

Advantage-Weighted Regression Methods. DPMPC is also related to advantage-weighted regression methods such as AWR (Peng et al., 2019) and AWAC (Nair et al., 2020), which update a policy by fitting actions from a replay buffer with weights determined by their estimated advantages. This connection is mainly at the level of the policy-improvement objective: both AWAC and DPMPC can be viewed as projecting an advantage-weighted target distribution onto a parameterized policy. However, the source and role of the supervised targets are different. AWAC learns from actions contained in an offline dataset or replay buffer, whereas DPMPC trains a diffusion prior from planner-generated action distributions that are stored in the replay buffer and periodically refreshed through generalized lazy reanalyze. Thus, DPMPC does not use advantage weighting for offline policy extraction; instead, it uses it to prioritize higher-value planner targets during policy training.

6 Conclusion

We introduce DPMPC, a planning-driven MBRL framework in which a diffusion policy serves as an expressive action prior for MPC. Trained on the planner distribution estimated from generalized lazy reanalyze, the diffusion prior reduces the actor divergence, so that the value function learned under the prior provides a more accurate estimate of the planner's return. Across DMControl, MyoSuite, and HumanoidBench, DPMPC achieves state-of-the-art asymptotic performance and improved sample efficiency, with the most pronounced gains on HumanoidBench. Value estimation analysis shows that DPMPC achieves the lowest value error. These results establish policy-prior parameterization as an important design factor in planning-driven MBRL. Future work may explore other expressive policy representations. Faster generative models, such as consistency models and flow-matching models, could reduce the training and inference cost while maintaining performance. These models might also be used as expressive priors to speed up the training of our framework.

AI USE STATEMENT

In this work, we used generative AI tools to assist in writing the derivation in Appendix A, to implement parts of the proposed method, and to support the analysis of experimental data. We have not used generative AI tools to formulate mathematical claims, to provide critical ingredients for their proofs, to propose or refine hypotheses, to help develop theoretical models or conceptual frameworks, to design or provide feedback on the research methodology or experiments, to interpret results, or to assist with translation. Generating synthetic datasets and cleaning or reformatting datasets are not applicable to this work. Additionally, we used generative AI tools to create or modify some of the figures reporting experimental results, to create and edit software code, to draft parts of the manuscript, to summarize and analyze existing literature, to search for relevant information, to edit the paper for readability, and to suggest the structure of the paper. We have reviewed all AI-assisted work: AI-generated code was verified and tested by the authors, and AI-assisted text and derivations were checked for correctness and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 specify the complete DPMPC training procedure. Appendix B lists the benchmark tasks, the implementation details, and all hyperparameters. It also identifies the official implementations and published results used for each baseline. All reported results are averaged over four random seeds per task. The full codebase for reproducing all reported experiments will be open-sourced upon acceptance.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. Myosuite: A contact-rich simulation suite for musculoskeletal motor control. In Roya Firoozi, Negar Mehr, Esen Yel, Rika Antonova, Jeannette Bohg, Mac Schwager, and Mykel Kochenderfer (eds.), *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pp. 492–507. PMLR, 23–24 Jun 2022. URL https://proceedings.mlr.press/v168/caggiano22a.html.

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.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=CpjKXe9rY7.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 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*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Christian Lagemann, Sajeda Mokbel, Miro Gondrum, Mario Rüttgers, Yuning Wang, Pol Suárez, Ludger Paehler, Deniz A Bezgin, Aaron B Buhendwa, Jared L Callaham, et al. The hydrogym reinforcement learning platform for fluid dynamics. *Nature*, pp. 1–8, 2026.

Jiaqiao Liang, Gaoming Lou, Fobao Zhou, Yumeng Cai, Chuang Wang, and Yitong Zhou. Residual reinforcement learning based on inverse kinematic modeling for soft robotic arm control. *Journal of Dynamic Systems, Measurement, and Control*, 147(6):061007, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6Anv3KB9lz.

Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning*, pp. 41163–41182. PMLR, 2024.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023.

Hikaru Sasaki and Takamitsu Matsubara. Variational policy search using sparse gaussian process priors for learning multimodal optimal actions. *Neural Networks*, 143:291–302, 2021.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoid-Bench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=1GNV9SW95eJ.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, October 2020. URL https://arxiv.org/abs/2010.02502.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

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.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=i7jAYFYDcM.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

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

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.

Minghui Ye and Ahmed H Elsheikh. Model-based reinforcement learning for active flow control. *Physics of Fluids*, 37(9), 2025.

Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. *Advances in Neural Information Processing Systems*, 2025.

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

A Derivation of the Advantage-Weighted Denoising Score-Matching Objective

We derive the advantage-weighted score-matching objective underlying Eq. 7 from the constrained policy optimization problem in Eq. 6. We first solve the KL-regularized optimization problem in closed form, which yields an optimal but intractable target policy. We then project this target onto the parameterized diffusion policy and replace the intractable log-likelihood of the diffusion model with its variational lower bound, which leads to the advantage-weighted score-matching objective.

Problem restatement. Let $\pi_{\text{lazy}}(\cdot|\mathbf{z})$ denote the lazily refreshed MPC target distribution stored in the replay buffer $\mathcal{B}$, let $d_{\mathcal{B}}(\mathbf{z})$ denote the latent-state distribution induced by $\mathcal{B}$, and let $A(\mathbf{z}, \mathbf{a})$ denote the advantage defined in Section 2.1. Recalling Eq. 6, we seek a diffusion policy $p_\theta(\mathbf{a}|\mathbf{z})$ that maximizes the expected advantage while staying close to π_{lazy} at every latent state:

$$\arg\max_{p_\theta} \int_{\mathbf{z}} d_{\mathcal{B}}(\mathbf{z}) \int_{\mathbf{a}} p_\theta(\mathbf{a}|\mathbf{z})\, A(\mathbf{z}, \mathbf{a})\, d\mathbf{a}\, d\mathbf{z}, \quad \text{s.t. } D_{\text{KL}}(p_\theta(\cdot|\mathbf{z}) \,\|\, \pi_{\text{lazy}}(\cdot|\mathbf{z})) \le \epsilon, \ \ \forall\, \mathbf{z}. \tag{8}$$

Optimal policy under KL regularization. Enforcing the pointwise KL constraint in Eq. 8 at every latent state would require a separate Lagrange multiplier per state, which is intractable. Following Peng et al. (2019), we therefore relax it to a single KL bound in expectation over the replay-buffer state distribution,

$$\int_{\mathbf{z}} d_{\mathcal{B}}(\mathbf{z})\, D_{\text{KL}}(p(\cdot|\mathbf{z}) \,\|\, \pi_{\text{lazy}}(\cdot|\mathbf{z}))\, d\mathbf{z} \ \le \ \epsilon, \tag{9}$$

which is controlled by a single temperature rather than a per-state one. Introducing a Lagrange multiplier η ($\eta > 0$) for this soft KL constraint and per-state multipliers $\alpha_{\mathbf{z}}$ for the normalization constraint $\int_{\mathbf{a}} p(\mathbf{a}|\mathbf{z})\, d\mathbf{a} = 1$ yields the Lagrangian (Peng et al., 2019; Peters et al., 2010)

$$\begin{aligned} \mathcal{L}(p, \eta, \alpha) = &\int_{\mathbf{z}} d_{\mathcal{B}}(\mathbf{z}) \int_{\mathbf{a}} p(\mathbf{a}|\mathbf{z})\, A(\mathbf{z}, \mathbf{a})\, d\mathbf{a}\, d\mathbf{z} \\ &+ \eta \left(\epsilon - \int_{\mathbf{z}} d_{\mathcal{B}}(\mathbf{z})\, D_{\text{KL}}(p(\cdot|\mathbf{z}) \,\|\, \pi_{\text{lazy}}(\cdot|\mathbf{z}))\, d\mathbf{z} \right) \\ &+ \int_{\mathbf{z}} \alpha_{\mathbf{z}} \left(1 - \int_{\mathbf{a}} p(\mathbf{a}|\mathbf{z})\, d\mathbf{a} \right) d\mathbf{z}. \end{aligned} \tag{10}$$

Taking the functional derivative with respect to $p(\mathbf{a}|\mathbf{z})$ and setting it to zero gives

$$\frac{\delta \mathcal{L}}{\delta p(\mathbf{a}|\mathbf{z})} = d_{\mathcal{B}}(\mathbf{z})\, A(\mathbf{z}, \mathbf{a}) - \eta\, d_{\mathcal{B}}(\mathbf{z}) \left(\log \frac{p(\mathbf{a}|\mathbf{z})}{\pi_{\text{lazy}}(\mathbf{a}|\mathbf{z})} + 1 \right) - \alpha_{\mathbf{z}} = 0. \tag{11}$$

Dividing through by $\eta\, d_{\mathcal{B}}(\mathbf{z})$ and rearranging yields

$$\log p(\mathbf{a}|\mathbf{z}) = \log \pi_{\text{lazy}}(\mathbf{a}|\mathbf{z}) + \frac{A(\mathbf{z}, \mathbf{a})}{\eta} - 1 - \frac{\alpha_{\mathbf{z}}}{\eta\, d_{\mathcal{B}}(\mathbf{z})}. \tag{12}$$

Exponentiating both sides gives

$$p(\mathbf{a}|\mathbf{z}) = \pi_{\text{lazy}}(\mathbf{a}|\mathbf{z})\, \exp\left(\frac{A(\mathbf{z}, \mathbf{a})}{\eta} \right) \exp\left(-\frac{\alpha_{\mathbf{z}}}{\eta\, d_{\mathcal{B}}(\mathbf{z})} - 1 \right). \tag{13}$$

The second exponential factor depends on $\mathbf{z}$ but is independent of $\mathbf{a}$. Therefore, for a fixed latent state, it is constant across all actions and can be interpreted as a $\mathbf{z}$-dependent normalizing factor $1/Z(\mathbf{z})$. Enforcing the normalization condition $\int_{\mathbf{a}} p(\mathbf{a}|\mathbf{z})\, d\mathbf{a} = 1$ in Eq. 13 determines its value:

$$Z(\mathbf{z}) = \exp\left(\frac{\alpha_{\mathbf{z}}}{\eta\, d_{\mathcal{B}}(\mathbf{z})} + 1 \right) = \int_{\mathbf{a}'} \pi_{\text{lazy}}(\mathbf{a}'|\mathbf{z})\, \exp\left(\frac{A(\mathbf{z}, \mathbf{a}')}{\eta} \right) d\mathbf{a}'. \tag{14}$$

The optimal policy is therefore

$$p^*(\mathbf{a}|\mathbf{z}) = \frac{1}{Z(\mathbf{z})}\, \pi_{\text{lazy}}(\mathbf{a}|\mathbf{z})\, \exp\left(\frac{A(\mathbf{z}, \mathbf{a})}{\eta} \right). \tag{15}$$

Advantage-weighted score-matching objective. The optimal policy p^* in Eq. 15 is defined only implicitly through the advantage function and cannot be directly represented by the diffusion model. We therefore project p^* onto the parameterized diffusion policy p_θ by minimizing the forward KL divergence (Peng et al., 2019), weighted by the same state distribution $d_\mathcal{B}$ inherited from Eq. 8:

$$\min_\theta \ \mathbb{E}_{\mathbf{z}\sim d_\mathcal{B}}[D_{\mathrm{KL}}(p^*(\cdot|\mathbf{z}) \,\|\, p_\theta(\cdot|\mathbf{z}))] \,. \tag{16}$$

Expanding the KL divergence gives

$$\min_\theta \ \mathbb{E}_{\mathbf{z}\sim d_\mathcal{B}}\left[\mathbb{E}_{\mathbf{a}\sim p^*(\cdot|\mathbf{z})}[\log p^*(\mathbf{a}|\mathbf{z}) - \log p_\theta(\mathbf{a}|\mathbf{z})]\right] . \tag{17}$$

The first term is independent of θ, so the problem is equivalently written as

$$\max_\theta \ \mathbb{E}_{\mathbf{z}\sim d_\mathcal{B}}\left[\mathbb{E}_{\mathbf{a}\sim p^*(\cdot|\mathbf{z})}[\log p_\theta(\mathbf{a}|\mathbf{z})]\right] . \tag{18}$$

To avoid sampling from the intractable policy p^*, we rewrite the inner expectation using π_{lazy} as the proposal distribution, while the outer expectation remains over the replay-buffer state distribution $\mathbf{z} \sim \mathcal{B}$:

$$\max_\theta \ \mathbb{E}_{\mathbf{z}\sim\mathcal{B},\, \mathbf{a}_0\sim\pi_{\text{lazy}}(\cdot|\mathbf{z})}\left[\frac{p^*(\mathbf{a}_0|\mathbf{z})}{\pi_{\text{lazy}}(\mathbf{a}_0|\mathbf{z})} \log p_\theta(\mathbf{a}_0|\mathbf{z})\right] . \tag{19}$$

Substituting Eq. 15 into the above equation yields

$$\max_\theta \ \mathbb{E}_{\mathbf{z}\sim\mathcal{B},\, \mathbf{a}_0\sim\pi_{\text{lazy}}(\cdot|\mathbf{z})}\left[\frac{1}{Z(\mathbf{z})} \exp\left(\frac{A(\mathbf{z},\mathbf{a}_0)}{\eta}\right) \log p_\theta(\mathbf{a}_0|\mathbf{z})\right] . \tag{20}$$

Since $Z(\mathbf{z}) > 0$ does not depend on θ, it is dropped from the optimization for simplicity (Peng et al., 2019), giving the advantage-weighted maximum-likelihood objective:

$$\max_\theta \ \mathbb{E}_{\mathbf{z}\sim\mathcal{B},\, \mathbf{a}_0\sim\pi_{\text{lazy}}(\cdot|\mathbf{z})}[w(\mathbf{z},\mathbf{a}_0)\, \log p_\theta(\mathbf{a}_0|\mathbf{z})] \,, \quad w(\mathbf{z},\mathbf{a}_0) = \exp\left(\frac{A(\mathbf{z},\mathbf{a}_0)}{\eta}\right) . \tag{21}$$

When p_θ is parameterized as a diffusion policy, the marginal log-likelihood $\log p_\theta(\mathbf{a}_0|\mathbf{z})$ becomes intractable. Following the DDPM framework (Ho et al., 2020) (see Section 2.2), we instead optimize its evidence lower bound (ELBO), whose simplified form reduces to a standard noise-prediction objective. Since the advantage weight $w(\mathbf{z},\mathbf{a}_0) \geq 0$ depends only on the latent state and the clean action, and is independent of both the diffusion timestep τ and the sampled noise $\boldsymbol{\epsilon}$, it can be taken outside the denoising expectation. Applying this reweighting to Eq. 21, and replacing the theoretical advantage A with its bootstrap estimate $\hat{A}$ from Section 3.2, yields the following advantage-weighted score-matching objective:

$$\mathcal{L}_{\text{awr}}(\theta) = \mathbb{E}_{\substack{\tau\sim\mathrm{Unif}\{1,\dots,T\},\ \mathbf{z}\sim\mathcal{B},\ \mathbf{a}_0\sim\pi_{\text{lazy}}(\cdot|\mathbf{z}),\\ \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}} \left[w(\mathbf{z},\mathbf{a}_0)\, \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\sqrt{\bar{\alpha}_\tau}\,\mathbf{a}_0 + \sqrt{1-\bar{\alpha}_\tau}\,\boldsymbol{\epsilon},\ \tau,\ \mathbf{z}\big)\right\|^2\right] ,$$
$$w(\mathbf{z},\mathbf{a}_0) = \exp\left(\frac{\hat{A}(\mathbf{z},\mathbf{a}_0)}{\eta}\right) . \tag{22}$$

Applying Eq. 22 along H-step latent rollouts with temporal weighting λ^t, and normalizing the advantage by the running scale S, yields the final DPMPC loss in Eq. 7.

B Experimental Details

B.1 Benchmarks

We evaluate on 14 challenging high-dimensional continuous control tasks spanning three benchmark suites: the Dog-Run, Dog-Walk, Dog-Trot, and Dog-Stand environments from DMControl (Tassa et al., 2018); the Key-Turn-Hard, Pen-Twirl-Hard, and Obj-Hold-Hard dexterous manipulation environments from MyoSuite (Caggiano et al., 2022); and the H1hand-Run, H1hand-Walk, H1hand-Slide, H1hand-Sit-Hard, H1hand-Stand, H1hand-Pole, and H1hand-Hurdle locomotion environments from HumanoidBench (Sferrazza et al., 2024). The observation and action space dimensions of the respective environments are shown in Table 4.

Table 4: **Environment specifications.** Observation and action space dimensions for all evaluation environments.

Environment	Obs. Dim.	Act. Dim.
Dog-Run	223	38
Dog-Walk	223	38
Dog-Trot	223	38
Dog-Stand	223	38
Key-Turn-Hard	93	39
Pen-Twirl-Hard	83	39
Obj-Hold-Hard	91	39
H1hand-Run	151	61
H1hand-Walk	151	61
H1hand-Slide	151	61
H1hand-Sit-Hard	151	61
H1hand-Stand	151	61
H1hand-Pole	151	61
H1hand-Hurdle	151	61

B.2 Implementation Details and Hyperparameters

Our implementation is based on the TD-MPC2 (Hansen et al., 2023) and BMPC (Wang et al., 2025) codebases. For the shared world model components (encoder, latent dynamics, reward predictor, value function, and MPPI planner), we adopt the TD-MPC2 hyperparameter settings without additional tuning and use the same configuration across all tasks. For the value function, we model the state value instead of the state-action value to reduce the number of parameters. Table 5 lists the hyperparameters for diffusion policy training introduced in DPMPC. These values are shared across all tasks.

All DPMPC results are averaged over four random seeds per task. For the BMPC and BOOM baselines, we train and evaluate using their respective official implementations[1]. For SDAC, we adapt the code from its official repository[2] to each task and run it ourselves to obtain the training curves. For SAC, TD-MPC2, and DreamerV3, we directly adopt the published results from the official TD-MPC2 benchmark[3] on the DMControl and MyoSuite tasks, and from the official HumanoidBench benchmark[4] on the H1hand tasks.

Table 5: **Diffusion prior hyperparameters.** All values are shared across tasks.

Hyperparameter	Value
Diffusion steps (T)	20
No. of hidden layers	2
Hidden dimensions	256
Activation	Mish
Time embedding dimension	32
Learning rate	3×10^{-4}
Advantage-based reweighting clamp range	$[0, 50]$
Reanalyze interval (K)	10
Reanalyze batch size (B_r)	20
Number of MPC realizations per state (M)	2
Advantage temperature (η)	0.5
Planning horizon (H)	3
Planner action samples per state	20
Update-to-data ratio	1

[1] BMPC codebase: `https://github.com/wertyuilife2/bmpc`; BOOM codebase: `https://github.com/molumitu/BOOM_MBRL`.

[2] SDAC codebase: `https://github.com/mahaitongdae/diffusion_policy_online_rl`.

[3] Baseline results and codebase available at `https://github.com/nicklashansen/tdmpc2`.

[4] Baseline results obtained from `https://github.com/carlosferrazza/humanoid-bench`.

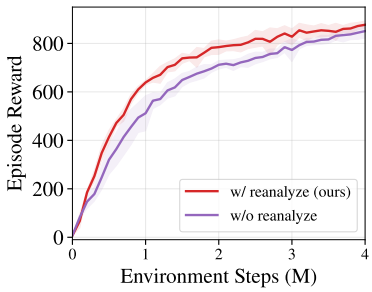


Figure 6: **Ablation of generalized lazy reanalyze on Dog-Run.** Disabling the periodic refresh of the stored π_{lazy} targets slows learning.

B.3 Value Estimation Error Protocol

For each checkpoint, we execute the planner and record 50 states along its trajectories at intervals of 20 environment steps, resetting the environment whenever an episode terminates. Throughout, the planner runs in exploration mode: each executed action is sampled from the optimized Gaussian $\mathcal{N}(\boldsymbol{\mu}_0^*, (\boldsymbol{\sigma}_0^*)^2)$ rather than set to its mean. For each recorded state, we reset the simulator to that state and follow the planner until episode termination, computing the discounted return. We repeat this rollout 50 times per state and average the returns to obtain the reference value G^π. We report the relative mean absolute error $100 \sum_i |\hat{V}(\mathbf{s}_i) - G^\pi(\mathbf{s}_i)| / \sum_i G^\pi(\mathbf{s}_i)$ over the recorded states.

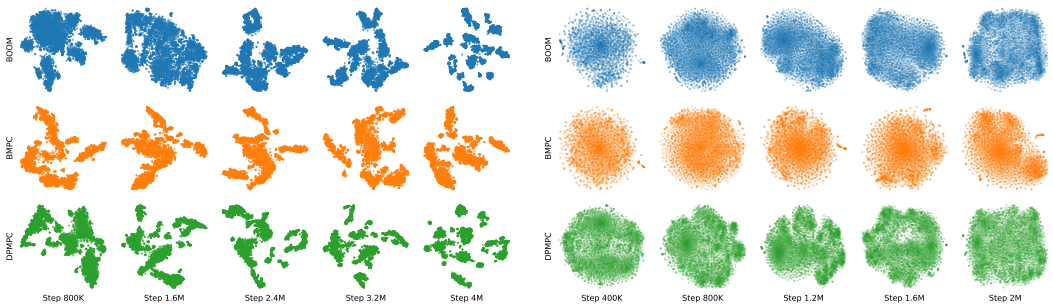
C Additional Ablation Studies

Effect of generalized lazy reanalyze. Figure 6 isolates the contribution of generalized lazy reanalyze by disabling the periodic refresh, so that each stored π_{lazy} target is never recomputed after its first MPC call. Without the refresh, learning is markedly slower. This indicates that periodically rerunning MPC is necessary to keep the estimate of the planner distribution up to date.

D Additional Visualizations

State visitation on Dog-Run and H1hand-Run. We examine what a diffusion policy prior changes about the planner's behavior. We compare the state visitation distributions of DPMPC, BMPC, and BOOM on Dog-Run and H1hand-Run throughout training. BMPC and BOOM use Gaussian policy priors, whereas DPMPC uses a diffusion policy prior that can represent richer action distributions. We evaluate five evenly spaced checkpoints and, at each checkpoint, collect 10 rollout episodes before applying t-SNE to project the visited states into two dimensions. Figure 7a shows that on Dog-Run, DPMPC covers a substantially broader region of the projected state space during the early training phase, even though all three methods achieve comparable episodic returns. This behavior is consistent with the diffusion prior providing more diverse candidate actions to the planner. As training progresses, the state visitation distributions of DPMPC and BOOM grow more similar, while BMPC converges to a markedly less diverse region of the state space—consistent with its comparatively weaker asymptotic performance. Figure 7b provides the corresponding t-SNE visualization on H1hand-Run. Consistent with our Dog-Run findings, the low-dimensional state distribution exhibits multiple modes throughout training. The number of modes is noticeably larger than for the two baselines, suggesting more diverse behavior under DPMPC.

Diffusion policy action distribution. Figures 8 and 9 visualize the per-dimension action distributions sampled from the learned diffusion policy on Dog-Run (checkpoint at 2M steps) and H1hand-Run (checkpoint at 1M steps), respectively. Unlike a Gaussian policy, which constrains each action dimension to a unimodal distribution, the diffusion policy captures multimodal and non-Gaussian structure. This expressiveness is evident across both tasks: many dimensions exhibit clearly separated modes or heavy-tailed shapes that a Gaussian prior could not represent, providing a richer set of candidate actions for the model-based planning stage.



(a) (b)

Figure 7: **State visitation on Dog-Run (a) and H1hand-Run (b).** t-SNE visualization of state visitation distributions across training for DPMPC, BMPC, and BOOM (5 checkpoints, 10 rollout episodes each). DPMPC visits a substantially more diverse set of states on Dog-Run and H1hand-Run.

E Training Time Comparison

Table 2 reports the wall-clock training time on the Dog-Run and H1hand-Run tasks over 2M environment steps. Dog-Run uses an action repeat of 2, and H1hand-Run uses an action repeat of 1. All methods are benchmarked on the same hardware: a single NVIDIA RTX 3090 GPU with an AMD Ryzen Threadripper 3970X CPU, and each method uses a single GPU for training. The results show a moderate increase in training time compared with TD-MPC2 and BMPC, while DPMPC remains faster than BOOM. This overhead could be reduced by faster samplers (Song et al., 2020) and alternative generative-policy parameterizations, such as consistency models (Song et al., 2023) or flow-matching models (Lipman et al., 2022).

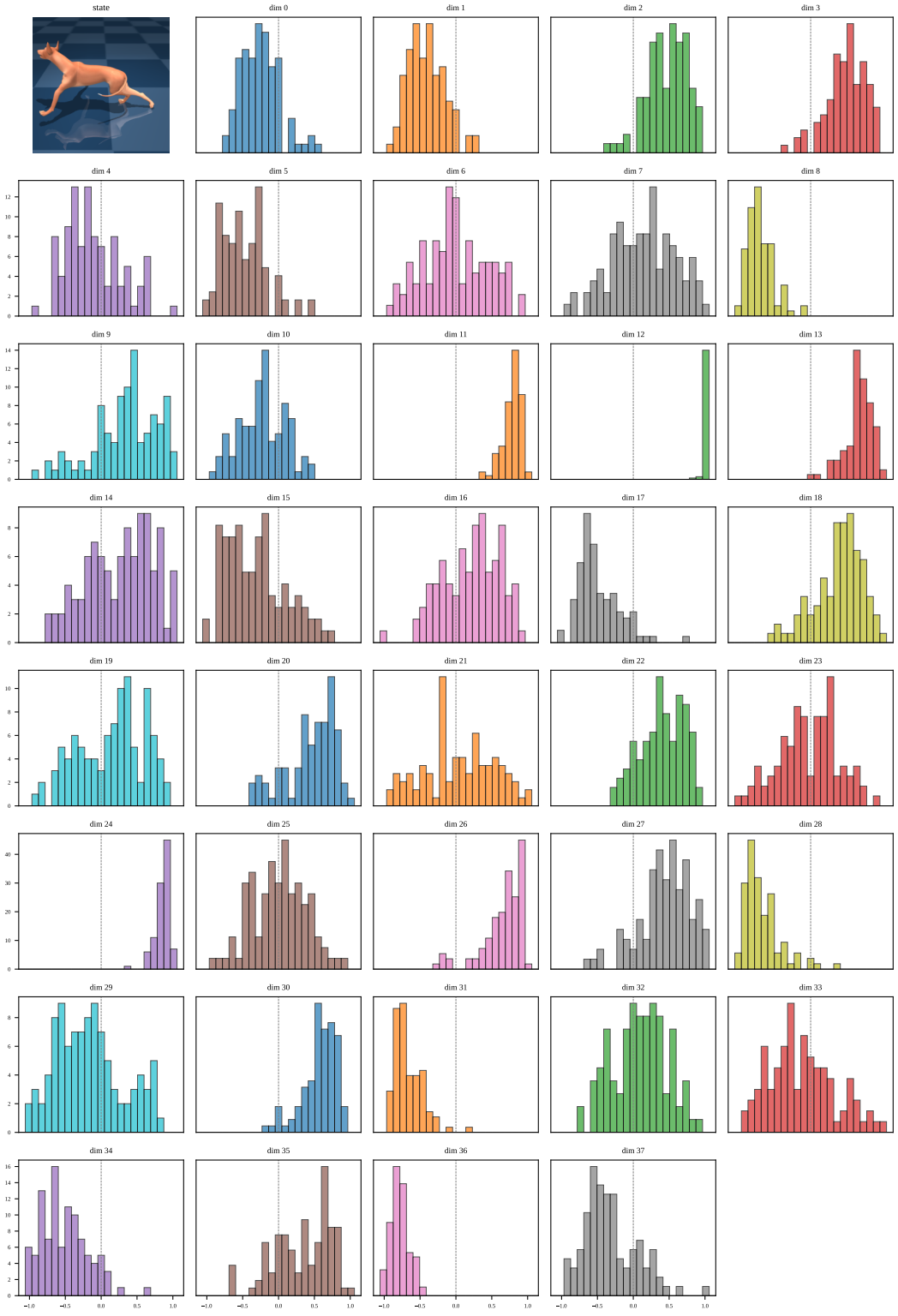


Figure 8: **Action distributions on Dog-Run.** Per-dimension action histograms sampled from the learned diffusion policy (2M steps). Multiple dimensions display multimodal or non-Gaussian distributions, demonstrating the expressiveness of the diffusion policy prior.



Figure 9: **Action distributions on H1hand-Run.** Per-dimension action histograms sampled from the learned diffusion policy (1M steps).