MODEL PREDICTIVE CONTROL WITH DIFFERENTIABLE WORLD MODELS FOR OFFLINE REINFORCEMENT LEARNING

Rohan Deb
Siebel School of Computing and Data Science,
University of Illinois Urbana-Champaign,
rd22@illinois.edu

Stephen J. Wright
Department of Computer Sciences,
University of Wisconsin Madison,
sjwright2@wisc.edu

Arindam Banerjee
Siebel School of Computing and Data Science,
University of Illinois Urbana-Champaign,
arindamb@illinois.edu

ABSTRACT

Offline Reinforcement Learning (RL) aims to learn optimal policies from fixed offline datasets, without further interactions with the environment. Such methods train an offline policy (or value function), and apply it at inference time without further refinement. We introduce an inference time adaptation framework inspired by model predictive control (MPC) that utilizes a pretrained policy along with a learned world model of state transitions and rewards. While existing world model and diffusion-planning methods use learned dynamics to generate imagined trajectories during training, or to sample candidate plans at inference time, they do not use inference-time information to *optimize* the policy parameters on the fly. In contrast, our design is a *Differentiable World Model* (DWM) pipeline that enables end-to-end gradient computation through imagined rollouts for *policy optimization at inference time* based on MPC. We evaluate our algorithm on D4RL continuous-control benchmarks (MuJoCo locomotion tasks and AntMaze), and show that exploiting inference-time information to optimize the policy parameters yields consistent gains over strong offline RL baselines.

1 INTRODUCTION

Offline RL learns a policy in a Markov decision process from a fixed dataset of previously collected trajectories, without any additional interaction with the environment. This setting arises naturally in domains where data is abundant but online interaction is constrained. Examples include modern recommender and advertising systems, which operate on massive logs of user interactions; healthcare and other safety-critical decision problems, which cannot permit exploratory actions; robotics and control, which may be limited by real-world cost, wear, and safety; and autonomous driving and navigation, which rely typically on recorded datasets rather than unrestricted on-policy exploration.

The core challenge of offline RL is to optimize reward while avoiding actions that are unsupported by the data distribution (Lange et al., 2012; Levine et al., 2020). Offline RL algorithms address this challenge by shaping either the policy update or the value-learning objective: Behavior regularized actor critic (BRAC) approaches keep the learned policy close to the dataset behavior (Fujimoto and Gu, 2021; Tarasov et al., 2023b); conservative Q learning (CQL) penalizes high Q-values on unlikely actions to reduce overestimation under dataset shift (Kumar et al., 2020); and in-sample or implicit methods avoid querying out of distribution actions during improvement (e.g., IQL) (Kostrikov et al., 2022). Separately, motivated by the success of generative models, many recent works replace simple parametric policies with generative models that capture multi-modal behavior in the dataset. These include sequence-modeling policies that predict actions conditioned on return (Decision Transformer) (Chen et al., 2021) and diffusion or flow-based planners/policies that generate actions or trajectories by iterative denoising or flow dynamics (e.g., Diffuser, and recent flow-matching policy work) (Janner et al., 2022a; Lipman et al., 2023). For a detailed discussion on related works, see Appendix A.

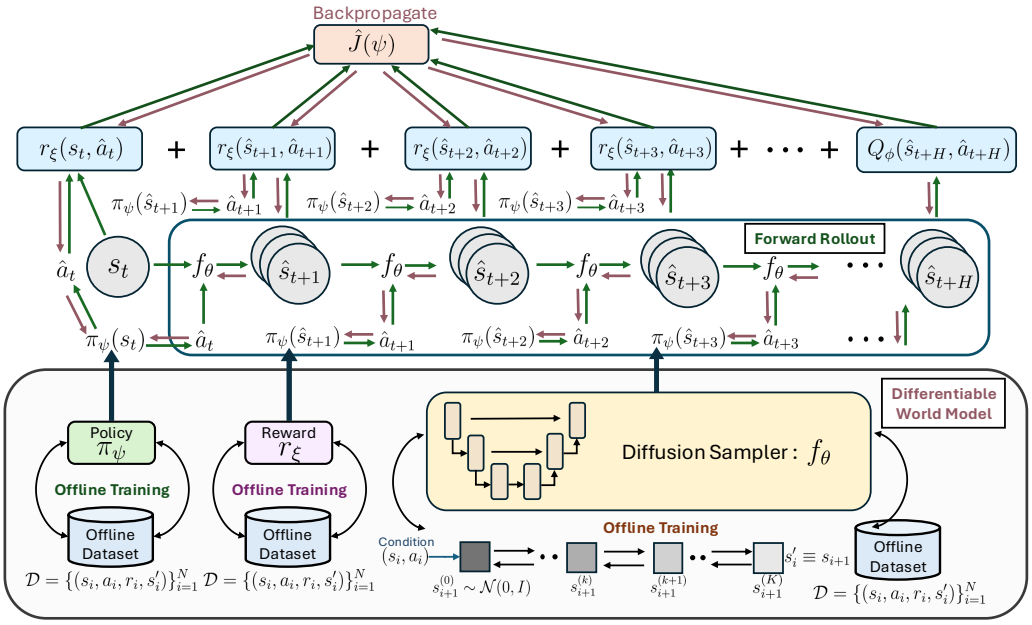

Figure 1: Inference-time MPC with a diffusion world model. An offline dataset is used to train (i) a policy π_ψ and terminal critic Q_ϕ (ii) a reward model r_ξ, and (iii) a diffusion-based dynamics sampler f_θ. At inference time, starting from the current state s_t, we unroll multiple imagined rollouts by alternating policy actions $\tilde{a}_h = \pi_\psi(\tilde{s}_h)$ and diffusion transitions $\tilde{s}_{h+1} = f_\theta(\tilde{s}_h, \tilde{a}_h, \varepsilon_{t+h})$, evaluate a finite-horizon surrogate return (predicted rewards plus terminal value), and backpropagate through the differentiable rollout to update ψ before executing the first action in the real environment. Green arrows indicate the forward rollout; red arrows indicate gradient flow.

Existing offline RL methods typically produce a single policy trained on the offline dataset and then deploy it "as-is" at inference time. These methods do not explicitly exploit inference-time information about the particular states encountered by the agent beyond the policy's standard state input. Specifically, an offline learner returns a policy and an associated Q-value, intended to approximate the optimal policy and Q-values (see Section 2.2 for definitions). When the policy is deployed without further adaptation, performance depends primarily on how accurate these approximations are in the states and actions *encountered at inference time*. We argue this is challenging for two reasons. First, estimating Q-value is intrinsically difficult because it represents a long-horizon quantity: it aggregates future rewards over many steps starting from a given state action pair. Second, in offline RL the critic must approximate the Q-value of the *optimal* policy, even though the available data might be generated by a different behavior policy.

In the Markovian model, the one-step transition dynamics $P(s' \mid s, a)$ and reward function $r(s, a)$ are local objects: they depend only on the current state-action pair, rather than on long-horizon estimates, and are independent of whatever policy is being deployed. This fact motivates a different approach: Learn world models of P and r from offline data and use these learned functions at inference time to formulate policy. The link between P and r and a corresponding optimal policy is provided by Model Predictive Control (MPC) (Camacho and Bordons, 2007; Rawlings et al., 2017). Conventional MPC optimizes action sequences over rolled-out short-horizon trajectories defined by the function P, thus implicitly generating a policy. We use MPC in a different way, using information gathered from the rollout simulations to update the parameters that define our policy. In this way, the initial policy learned from the offline data is updated continuously during deployment as new information is gathered about the system. We now describe our method in detail and outline our contributions.

1. We introduce a *Differentiable World Model* (DWM) pipeline that consists (i) a state-transition sampler, (ii) a reward model, and (iii) a terminal-value function for finite-horizon evaluation. A key design choice is to make these components *differentiable* w.r.t. their inputs or conditionings, so the entire pipeline forms a computation graph that supports gradient-based optimization. See Section 3 for a detailed description.

2. At inference time, we use the current state s_t to generate multiple imagined rollouts by unrolling the differentiable dynamics under the current policy. We score these rollouts with a surrogate objective built from predicted rewards and a terminal-value function. We then update the policy parameters with gradient-based steps before executing one step of the resulting action in the real environment. See Figure 1 for a visual overview and Section 4 for a detailed description.
3. We instantiate the differentiable state-transition sampler with a diffusion model and derive a policy update that backpropagates through imagined rollouts, expressing the gradient in terms of policy Jacobians and diffusion-dynamics Jacobians via a recursive chain rule (see Theorem 4.1).
4. We evaluate our algorithm on standard D4RL continuous control benchmarks (Fu et al., 2020), including MuJoCo locomotion tasks (18 datasets) as well as the more challenging AntMaze environments (6 datasets). We show that our approach of leveraging inference time optimization using imagined rollouts consistently outperforms strong offline RL baselines (see Section 5).

Comparison with existing World Model based Offline RL methods. Although world models (including diffusion-based world models) have been used in *offline* RL, they are typically leveraged in two ways. Some methods (Kidambi et al., 2020; Yu et al., 2020; 2021; Ding et al., 2024) use the learned dynamics primarily during *training*, generating imagined rollouts from the offline dataset to construct additional targets or synthetic experience for policy and value learning, including diffusion world models that model multi-step futures without step-by-step rollout. Other methods use a generative model at *inference time* to produce candidates for future optimal state trajectories via sampling (often with guidance via return-conditioning), and then select an action by executing the first action of a sampled plan, without adapting the underlying policy parameters during inference (Ajay et al., 2023; Janner et al., 2022a; Ki et al., 2025; Yun et al., 2024). In contrast, our method uses the current observed state to *adapt the policy parameters at inference time* by backpropagating through a differentiable world model over imagined finite-horizon rollouts.

2 Preliminaries

2.1 Offline Reinforcement Learning

We consider a discounted Markov decision process (MDP) specified by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, d_0, r, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $P(s' \mid s, a)$ is the transition kernel, d_0 is the initial state distribution, $r(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. A stochastic policy $\pi(a \mid s)$ induces a trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ with probability $p^\pi(\tau) = d_0(s_0) \prod_{t \geq 0} \pi(a_t \mid s_t) P(s_{t+1} \mid s_t, a_t)$ and discounted return $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

The RL objective is to find a return-maximizing policy $\pi^\star \in \operatorname{argmax}_\pi \mathbb{E}_{\tau \sim p^\pi}[R(\tau)]$.

Temporal-difference learning. TD methods approximate the optimal action-value function $Q^\star(s, a) = \mathbb{E}_{\tau \sim p^{\pi^\star}}[R(\tau) \mid s_0 = s, a_0 = a]$ with a parameterized critic Q_θ by minimizing a Bellman error on transition data: $\mathcal{L}_{\mathrm{TD}}(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\left(r + \gamma \max_{a' \in \mathcal{A}} Q_\theta(s', a') - Q_\theta(s, a) \right)^2 \right]$.
For continuous action spaces, the maximization is typically implemented via a parameterized actor $\pi_\phi(a \mid s)$, leading to a policy objective of the form $\mathcal{J}(\phi) := \mathbb{E}_{s \sim \mathcal{D},\, a \sim \pi_\phi(\cdot \mid s)}[Q_\theta(s, a)]$.

Offline RL and distribution shift. Offline RL learns a policy from a fixed dataset without additional environment interactions. We assume access to a static dataset of transitions $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ collected by an unknown behavior policy μ. A core difficulty is that naively applying TD learning and policy improvement can drive the learned policy π_ϕ toward state-action regions that are weakly represented in $\mathcal{D}$, i.e., the induced occupancy measure d^{π_ϕ} moves away from d^μ. Many offline RL methods address this distribution shift by enforcing an explicit constraint, for example of the form $D(d^{\pi_\phi} \,\|\, d^\mu) \leq \varepsilon$, where D is a divergence or discrepancy measure, incorporated directly into the learning procedure. Such constrained formulations often introduce additional algorithmic heuristics to obtain stable and competitive performance in practice.

2.2 Diffusion Probabilistic Models

Diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020) define a latent-variable generative model by reversing a fixed forward diffusion (noising) process. Let $x^0 \in \mathbb{R}^d$ denote a data sample. The model likelihood is defined by introducing latent variables $x^1, \ldots, x^K$ and marginalizing them out: $p_\theta(x^0) := \int p(x^K) \prod_{k=1}^K p_\theta(x^{k-1} \mid x^k)\, dx^{1:K}$, where $p(x^K)$ is typically a standard

Gaussian prior. The forward diffusion chain gradually corrupts the data by adding Gaussian noise according to a variance schedule $\{\beta_k\}_{k=1}^K$ i.e., $q(x^{1:K} \mid x^0) := \prod_{k=1}^K q(x^k \mid x^{k-1})$, with one-step transitions $q(x^k \mid x^{k-1}) := \mathcal{N}\big(x^k; \sqrt{1-\beta_k}\, x^{k-1}, \beta_k I\big)$.

The reverse (denoising) process is parameterized as a Gaussian with timestep-dependent covariance $p_\theta(x^{k-1} \mid x^k) = \mathcal{N}\big(x^{k-1}; \mu_\theta(x^k, k), \Sigma_k\big)$, where $\mu_\theta(x^k, k)$ is a neural network that maps the noisy latent x^k and timestep k to the mean of the reverse transition. Although a tractable variational lower bound on $\log p_\theta(x)$ can be optimized to train diffusion models, in practice it is common to use the simplified surrogate objective proposed by Ho et al. (2020), which trains the model to predict the injected noise. Specifically, the predefined forward noising process is $q(x^k \mid x^{k-1}) := \mathcal{N}\Big(x^k; \sqrt{\alpha_{k-1}}\, x^{k-1}, (1-\alpha_{k-1})I\Big)$, and the learned reverse process is parameterized as $p_\theta(x^{k-1} \mid x^k) := \mathcal{N}\Big(x^{k-1}; \mu_\theta(x^k, k), \Sigma_k\Big)$, where $\{\alpha_k\}_{k=0}^{K-1}$ specifies the variance schedule, $x^0 := x$ is a data sample, and $x^K \sim \mathcal{N}(0, I)$ for K large enough. The simplified denoising loss is

$$\mathcal{L}_{\text{denoise}}(\theta) := \mathbb{E}_{\substack{k\sim\text{Unif}(\{1,\dots,K\}) \\ x^0\sim q,\ \epsilon\sim\mathcal{N}(0,I)}}\Big[\big\|\epsilon - \epsilon_\theta(x^k, k)\big\|_2^2\Big].$$

where $\epsilon_\theta(x^k, k)$ is a neural network that predicts the Gaussian noise ϵ added to obtain the noisy latent x^k. This is equivalent to predicting the reverse-process mean, since $\mu_\theta(x^k, k)$ can be expressed as a function of $\epsilon_\theta(x^k, k)$ (Ho et al., 2020).

3 Components of the World Model

Our algorithm at inference time requires sampling next state samples conditioned on the current state action pair, and differentiating those samples with respect to the conditioning variables. We refer to any model with this sample plus gradient interface as a differentiable generative world model (DiffGenWM), and in this work we implement it using a conditional diffusion model. In addition to the DiffGenWM module, our world model includes a differentiable reward model and a terminal value function given by the critic of a pretrained policy. Concretely, we build a world model using the offline dataset of trajectories $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^N$, consisting of the following components.

1. A *Differentiable Diffusion Sampler* f_θ to simulate the transition dynamics.
2. A reward model r_ξ to learn the reward of a given state action pair.
3. A pretrained policy π_ψ and the corresponding pretrained critic value Q_ϕ.

Next we describe all these components in detail.

3.1 Differentiable Diffusion Sampler

Our objective is to learn a parametric differentiable diffusion sampler that conditioned on a given state-action pair s_t, a_t at time t and sampled noise ϵ_t generates the next state $s_{t+1} = f_\theta(s_t, a_t, \varepsilon_t), \varepsilon_t \sim p_0(\varepsilon)$. Here f_θ is the *reverse diffusion sampler*, written as a deterministic computation graph, and ε_t is a collection of Gaussian random variables used in the generation process. Equivalently, the diffusion model specifies a conditional distribution over next states, $p_\theta(s_{t+1} \mid s_t, a_t)$, together with a reparameterized sampling procedure $s_{t+1} = f_\theta(s_t, a_t, \varepsilon_t)$ whose randomness is isolated in ε_t.

We learn $p_\theta(s_{t+1} \mid s_t, a_t)$ from an offline dataset of transitions $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^N$. The diffusion model is trained to represent the conditional law of s_{t+1} given (s_t, a_t) by introducing a *forward noising process* on s_{t+1} and a learned *reverse denoising process* that inverts this noising when conditioned on (s_t, a_t). At sampling time, the reverse process induces the map f_θ.

3.1.1 Forward process

Fix a diffusion horizon $K \in \mathbb{N}$ and a variance schedule $\{\alpha_k\}_{k=1}^K \subset (0, 1)$. For each transition tuple $(s_t, a_t, s_{t+1}) \in \mathcal{D}$, define a Markovian forward process that progressively corrupts the next state as:

$$s_{t+1}^{(0)} = s_{t+1}, \text{ and } s_{t+1}^{(k)} \mid s_{t+1}^{(k-1)} \sim q\big(s^{(k)} \mid s^{(k-1)}\big) = \mathcal{N}\big(\sqrt{\alpha_k}\, s_{t+1}^{(k-1)}, (1-\alpha_k)I\big), k = 1, \dots, K.$$

Let $\bar{\alpha}_k := \prod_{j=1}^k \alpha_j$. This choice implies a closed-form marginal for any noise level k given by $q\big(s_{t+1}^{(k)} \mid s_{t+1}\big) = \mathcal{N}\big(\sqrt{\bar{\alpha}_k}\, s_{t+1}, (1-\bar{\alpha}_k)I\big)$. In particular, one can write a reparameterized sample from the marginal as $s_{t+1}^{(k)} = \sqrt{\bar{\alpha}_k}\, s_{t+1} + \sqrt{1-\bar{\alpha}_k}\, \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$.

3.1.2 CONDITIONAL REVERSE PROCESS

The reverse process aims to sample s_{t+1} conditioned on (s_t, a_t) by iteratively denoising from a Gaussian reference distribution at level K. Let the conditioning be $c_t := (s_t, a_t)$. At sampling time, initialize from a base noise variable $s_{t+1}^{(K)} \sim \mathcal{N}(0, I)$, and apply a learned reverse transition for $k = K-1, \ldots, 1$:

$$s_{t+1}^{(k-1)} = \frac{1}{\sqrt{\alpha_k}}\Big(s_{t+1}^{(k)} - (1-\alpha_k)\,\hat{\epsilon}_\theta\big(s_{t+1}^{(k)}, k, c_t\big)\Big) + \sigma_k z_{k-1},\ \ z_{k-1} \sim \mathcal{N}(0, I).$$

Here $\hat{\epsilon}_\theta(\cdot, k, c_t)$ is a parametric predictor of the noise component at level k, and $\{\sigma_k\}$ specifies the reverse-process variance. The final denoised sample is $s_{t+1}^{(0)} \sim p_\theta(\cdot \mid s_t, a_t)$, and we define the reverse sampler f_θ by collecting all Gaussian random variables used by the reverse procedure into $\varepsilon_t := (z_K, z_{K-1}, \ldots, z_0)$, so that the sampled next state can be written as $s_{t+1}^{(0)} = f_\theta(s_t, a_t, \varepsilon_t)$.

We define initialization map $g_K : \mathbb{R}^d \to \mathbb{R}^d$ and reverse-step map $h_k : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^d$:

$$\begin{aligned} g_K(z) &:= z, \\ h_k(u, s, a, z) &:= \frac{1}{\sqrt{\alpha_k}}\Big(u - (1-\alpha_k)\,\hat{\epsilon}_\theta\big(u, k, (s, a)\big)\Big) + \sigma_k z, \quad \text{for } k = 1, \ldots, K-1 \end{aligned} \tag{1}$$

so that $s_{t+1}^{(k-1)} = h_k(s_{t+1}^{(k)}, s_t, a_t, z_{k-1})$ and $s_{t+1}^{(K)} = g_K(z_K)$.

Deterministic computation graph for fixed noise. For any fixed realization of ε_t, the mapping $(s_t, a_t) \mapsto s_{t+1}^{(0)}$ is a deterministic composition of (i) linear operations, (ii) evaluations of the denoiser $\hat{\epsilon}_\theta(\cdot, k, c_t)$ at each reverse step, and (iii) additive terms determined by the fixed Gaussian draws. Consequently, f_θ is a differentiable computation graph in its inputs (s_t, a_t) for fixed ε_t.

3.1.3 LEARNING OBJECTIVE

We train $\hat{\epsilon}_\theta$ to invert the forward corruption of s_{t+1}, conditioned on (s_t, a_t). Using the marginal reparameterization at a randomly chosen diffusion level k, we form $s_{t+1}^{(k)} = \sqrt{\bar{\alpha}_k}\, s_{t+1} + \sqrt{1-\bar{\alpha}_k}\,\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, and minimize the conditional noise-prediction error

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}) \sim \mathcal{D} \\ k \sim \mathrm{Unif}(\{1, \ldots, K\}) \\ \epsilon \sim \mathcal{N}(0, I)}} \Big[\big\|\epsilon - \hat{\epsilon}_\theta\big(s_{t+1}^{(k)}, k, c_t\big)\big\|_2^2\Big].$$

Intuitively, this objective teaches the denoiser to recover the injected Gaussian noise at arbitrary noise levels using (s_t, a_t) as side information. After training, the resulting reverse process defines a conditional generative model $p_\theta(s_{t+1} \mid s_t, a_t)$, and its sampling procedure is the map $s_{t+1} = f_\theta(s_t, a_t, \varepsilon_t), \varepsilon_t \sim p_0(\varepsilon)$, which we treat as a learned, differentiable simulator of one-step dynamics.

3.2 REWARD MODEL

In addition to the transition model, we learn a parametric reward predictor from the same offline data. We parameterize the reward predictor as a function $r_\xi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which maps a state–action pair (s_t, a_t) to a scalar reward estimate. Given an offline dataset of transitions $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^N$, we fit r_ξ by supervised regression on observed rewards: $\min_\xi\ \mathcal{L}_r(\xi) := \mathbb{E}_{(s_t, a_t, r_t, \cdot) \sim \mathcal{D}}\Big[\big(r_\xi(s_t, a_t) - r_t\big)^2\Big]$.

After training, r_ξ serves as a differentiable reward oracle that can be queried at arbitrary (s, a) pairs produced by downstream planning or policy optimization, and can be combined with the learned diffusion transition model to form multi-step return objectives.

3.3 POLICY AND TERMINAL VALUE

Given an offline dataset $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^N$, we learn a parametric policy $\pi_\psi(s)$ and a parametric action-value function $Q_\phi(s, a)$. We adopt a Behavior-Regularized Actor Critic (BRAC) style objective from (Wu et al., 2019) and follow the practical architectural and training choices from ReBRAC (Tarasov et al., 2023a). The critic is trained by one-step temporal-difference regression with a target network $Q_{\bar{\phi}}$: $\mathcal{L}_Q(\phi) := \mathbb{E}_{\substack{(s, a, r, s') \sim \mathcal{D} \\ a' \sim \pi_\psi(\cdot \mid s')}}\Big[\big(Q_\phi(s, a) - r - \gamma Q_{\bar{\phi}}(s', a')\big)^2\Big]$. The actor is

Algorithm 1 MPCwDWM (Model Predictive Control with Differentiable World Model)

Require: Diffusion sampler f_θ, noise $\{\varepsilon^{(i)}_{t:t+H-1}\}_{i=1}^{M}$ reward model r_ξ, terminal critic Q_ϕ, policy π_ψ, horizon H, particles M, inner steps E, step size α, discount γ.

for $t = 1, 2, 3 \ldots, T$ **do**
  Observe environment state s_t
  for $e = 1, 2, \ldots, E$ **do**
    $\tilde{s}_0^{(i)} \leftarrow s_t \quad \forall\, i \in \{1, 2, \ldots, M\}$
    $J \leftarrow 0$
    for $j = 0, 1, \ldots, H-1$ **do**
      for $i = 1, 2, \ldots, M$ **do**
        $\tilde{a}_j^{(i)} \leftarrow \pi_\psi(\tilde{s}_j^{(i)})$ ▷ Action for imagined rollout
      end for
      $J \leftarrow J + \frac{1}{M} \sum_{i=1}^{M} \gamma^j \, r_\xi(\tilde{s}_j^{(i)}, \tilde{a}_j^{(i)})$
      for $i = 1, 2, \ldots, M$ **do**
        $\tilde{s}_{j+1}^{(i)} \leftarrow f_\theta(\tilde{s}_j^{(i)}, \tilde{a}_j^{(i)}, \varepsilon_{t+j}^{(i)})$ ▷ Next state
      end for
    end for
    $J \leftarrow J + \frac{1}{M} \sum_{i=1}^{M} \gamma^H \, Q_\phi(\tilde{s}_H^{(i)}, \pi_\psi(\tilde{s}_H^{(i)}))$
    $\psi \leftarrow \psi + \alpha \, \nabla_\psi J$ ▷ Policy Update
  end for
  Execute $a_t \leftarrow \pi_\psi(s_t)$ and observe (r_t, s_{t+1})
end for

trained to select actions with high value under Q_ϕ, while remaining close to the dataset behavior by a behavior-cloning regularizer that increases the likelihood of dataset actions under π_ψ: $\mathcal{L}_\pi(\psi) := \mathbb{E}_{\substack{(s,a)\sim\mathcal{D} \\ a^\pi \sim \pi_\psi(\cdot|s)}} \Big[-Q_\phi(s, a^\pi) \;-\; \alpha \log \pi_\psi(a \mid s) \Big]$, where $\alpha > 0$ controls the strength of behavior regularization and $\gamma \in (0, 1)$ is the discount factor. In practice, $Q_{\bar{\phi}}$ is maintained as a slowly updated copy of Q_ϕ to stabilize the bootstrapped target.

4 MPC WITH DIFFERENTIABLE WORLD MODEL

We now describe a receding-horizon model predictive control (MPC) procedure that uses the learned diffusion world model as a differentiable simulator. The world model is given by the reparameterized transition map $s_{t+1} = f_\theta(s_t, a_t, \varepsilon_t), \varepsilon_t \sim p_0(\varepsilon)$, where f_θ denotes the reverse diffusion sampler unrolled as a computation graph. We combine this dynamics model with a learned reward predictor $r_\xi(s, a)$ and a learned terminal critic $Q_\phi(s, a)$. The critic provides a terminal value for truncated rollouts, while r_ξ supplies the immediate reward along imagined trajectories at inference time.

4.1 RECEDING-HORIZON OBJECTIVE

At a real environment state s_t, MPC optimizes a horizon-H objective over imagined trajectories generated by f_θ. Given a fixed noise sequence $\varepsilon_{t:t+H-1} = (\varepsilon_t, \ldots, \varepsilon_{t+H-1})$ define the imagined rollout recursively by

$$\tilde{s}_0 = s_t, \;\; \tilde{a}_j = \pi_\psi(\tilde{s}_j), \;\; \tilde{s}_{j+1} = f_\theta(\tilde{s}_j, \tilde{a}_j, \varepsilon_{t+j}), \tag{2}$$

where $j = 0, \ldots, H-1$. We evaluate the finite-horizon return by summing predicted stage rewards and adding a terminal value given by the critic:

$$L(\psi; \varepsilon_{t:t+H-1}) = \sum_{j=0}^{H-1} \gamma^j \, r_\xi(\tilde{s}_j, \tilde{a}_j) + \gamma^H \, Q_\phi\Big(\tilde{s}_H, \pi_\psi(\tilde{s}_H)\Big). \tag{3}$$

We then define the MPC objective as the expectation of this return over diffusion noise: $J_t(\psi) = \mathbb{E}_{\varepsilon_{t:t+H-1}\sim p_0}\Big[L(\psi; \varepsilon_{t:t+H-1})\Big]$.

4.2 Monte Carlo Approximation

We approximate $J_t(\psi)$ with M i.i.d. noise sequences $\{\varepsilon^{(m)}_{t:t+H-1}\}_{m=1}^{M}$, where $\varepsilon^{(m)}_{t:t+H-1} \sim p_0$. For each m, generate an imagined rollout $\{\tilde{s}^{(m)}_j, \tilde{a}^{(m)}_j\}_{j=0}^{H}$ by

$$\tilde{s}^{(m)}_0 = s_t, \quad \tilde{a}^{(m)}_j = \pi_\psi(\tilde{s}^{(m)}_j), \quad \tilde{s}^{(m)}_{j+1} = f_\theta\Big(\tilde{s}^{(m)}_j, \tilde{a}^{(m)}_j, \varepsilon^{(m)}_{t+j}\Big).$$

The Monte Carlo estimator of the objective $J_t(\psi)$ is given by the following expression:
$\widehat{J}_t(\psi) = \frac{1}{M}\sum_{m=1}^{M}\Big[\sum_{j=0}^{H-1}\gamma^j\, r_\xi\big(\tilde{s}^{(m)}_j, \tilde{a}^{(m)}_j\big) + \gamma^H\, Q_\phi\Big(\tilde{s}^{(m)}_H, \pi_\psi(\tilde{s}^{(m)}_H)\Big)\Big]$.

4.3 Gradient-based optimization

At time t, we perform E steps of gradient ascent on $\widehat{J}_t(\psi)$ while holding f_θ, r_ξ, and Q_ϕ fixed:

$$\psi \leftarrow \psi + \alpha \nabla_\psi \widehat{J}_t(\psi), \qquad e = 1, \dots, E, \tag{4}$$

where $\alpha > 0$ is a step size. After the inner loop, MPC executes the first action in the real environment $a_t = \pi_\psi(s_t)$, observes (r_t, s_{t+1}), and repeats the procedure at the next state s_{t+1} (see Algorithm 1).

4.4 Gradient recursion through diffusion rollouts

The next theorem states a compact Jacobian recursion for the gradient of the per-noise objective $L(\psi; \varepsilon_{t:t+H-1})$ in (3) under the rollout dynamics in (2) (see Appendix B for proof). Define the rollout Jacobians, for $j = 0, \dots, H-1$,

$$\Pi_s(j) := \nabla_s \pi_\psi(s)\big|_{s=\tilde{s}_j}, \quad \Pi_\psi(j) := \nabla_\psi \pi_\psi(\tilde{s}_j),$$
$$F_s(j) := \nabla_s f_\theta(s, a, \varepsilon_{t+j})\big|_{s=\tilde{s}_j, a=\tilde{a}_j}, F_a(j) := \nabla_a f_\theta(s, a, \varepsilon_{t+j})\big|_{s=\tilde{s}_j, a=\tilde{a}_j}.$$

Let $G_j := \nabla_\psi \tilde{s}_j$ and $D_j := \nabla_\psi \tilde{a}_j$. Then $G_0 = 0$ and, for $j = 0, \dots, H-1$,

$$D_j = \Pi_s(j)\, G_j + \Pi_\psi(j), G_{j+1} = F_s(j)\, G_j + F_a(j)\, D_j.$$

Further define the gradient of the reward and the critic as:

$$r_s(j) := \nabla_s r_\xi(\tilde{s}_j, \tilde{a}_j), \quad r_a(j) := \nabla_a r_\xi(\tilde{s}_j, \tilde{a}_j), Q_s := \nabla_s Q_\phi(\tilde{s}_H, \tilde{a}_H), \quad Q_a := \nabla_a Q_\phi(\tilde{s}_H, \tilde{a}_H).$$

Theorem 4.1 (Gradient recursion). *Fix a time t, a horizon H, and a noise sequence $\varepsilon_{t:t+H-1}$, and let $\{(\tilde{s}_j, \tilde{a}_j)\}_{j=0}^{H}$ be defined by (2). Assume π_ψ is differentiable in ψ and its state input, and f_θ, r_ξ, and Q_ϕ are differentiable in their state and action arguments.*
Then the gradient of the per-noise return in (3) is

$$\nabla_\psi L(\psi; \varepsilon_{t:t+H-1}) = \sum_{j=0}^{H-1} \gamma^j \Big(r_s(j)\, G_j + r_a(j)\, D_j\Big) + \gamma^H \Big(Q_s\, G_H + Q_a\, D_H\Big).$$

Moreover, if f_θ is implemented by a reverse diffusion recursion as given by (1) then

$$\nabla_a f_\theta(s, a, \varepsilon) = A_0, \qquad \nabla_s f_\theta(s, a, \varepsilon) = B_0,$$

where $A_K = 0$, $B_K = 0$, and for $k = K, \dots, 1$,

$$A_{k-1} = \frac{\partial h_k}{\partial u}\, A_k + \frac{\partial h_k}{\partial a}, \qquad B_{k-1} = \frac{\partial h_k}{\partial u}\, B_k + \frac{\partial h_k}{\partial s},$$

with all partial derivatives evaluated at $(u, s, a, z) = (s^{(k)}, s, a, z_{k-1})$.

Remark 4.2. Theorem 4.1 provides an explicit chain-rule decomposition of $\nabla_\psi L(\psi; \varepsilon_{t:t+H-1})$ along the rollout (2). The sensitivities (G_j, D_j) propagate how changing ψ perturbs the imagined state action sequence, through the policy Jacobians $(\Pi_s(j), \Pi_\psi(j))$ and the diffusion-dynamics Jacobians $(F_s(j), F_a(j))$. The final expression aggregates these perturbations through the stage rewards and the terminal critic via $(r_s(j), r_a(j))$ and (Q_s, Q_a), yielding the gradient used in the update (4).

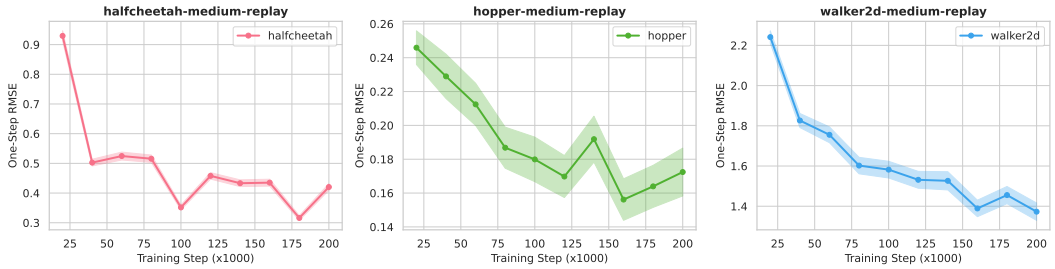

Figure 2: One-step state prediction RMSE of diffusion models across training steps (20k–200k) on medium-replay datasets. RMSE decreases with training for all three environments (halfcheetah, hopper, walker2d), with shaded regions showing standard error over 1000 transitions.

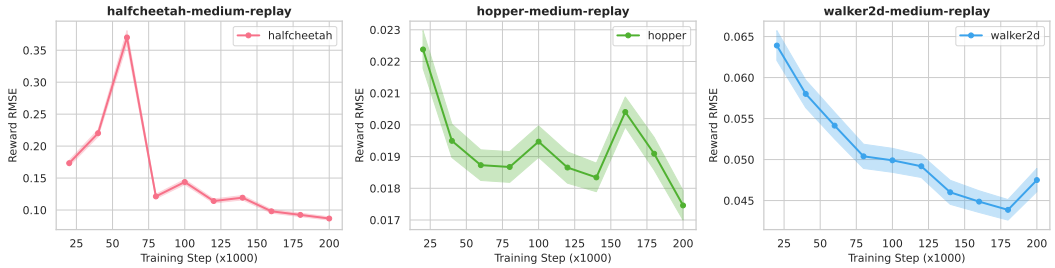

Figure 3: Reward prediction RMSE of reward models across training steps (20k–200k) on medium-replay datasets. All environments show decreasing RMSE with training. Shaded regions indicate standard error over 1000 transitions.

5 Experiments

Datasets. We evaluate on the D4RL benchmark (Fu et al., 2020), focusing on continuous-control domains where offline learning and long-horizon credit assignment are challenging. Our main results cover all Gym MuJoCo locomotion datasets (18 tasks spanning random, medium, expert, medium-expert, medium-replay, and full-replay settings across HalfCheetah, Hopper, and Walker2d) and all AntMaze datasets (6 tasks spanning maze, maze-diverse, medium-play, medium-diverse, large-play, and large-diverse). For each task, we report D4RL normalized scores computed from the return accumulated by the agent.

Baselines. We compare against strong ensemble-free offline RL baselines that are widely used on D4RL locomotion and AntMaze. These include TD3+BC (Fujimoto and Gu, 2021), which adds behavior cloning regularization to a TD3 actor update; IQL (Kostrikov et al., 2022), which performs offline policy improvement via value-weighted regression without explicit behavior constraints; CQL (Kumar et al., 2020), which trains a conservative critic by penalizing high values on out-of-distribution actions; SAC-RND (Nikulin et al., 2023), which augments SAC with random-network-distillation-based penalties to discourage out-of-distribution actions; and ReBRAC (Tarasov et al., 2023b), which revisits behavior-regularized actor-critic with practical improvements and strong performance. We also compare against generative policy and planning approaches in Appendix C.

Experimental details. Our approach has an offline training stage and an inference-time adaptation stage. Offline, we first train a policy π_ψ and critic Q_ϕ on the fixed dataset using ReBRAC (Tarasov et al., 2023b). We then train a diffusion transition model f_θ on dataset transitions and a reward predictor r_ξ on dataset rewards using supervised learning. Figure 2 reports the diffusion model prediction RMSE across training steps, and Figure 3 reports the reward model prediction RMSE across training steps. See Appendix C for the full diffusion and reward model results.

At inference time, for each real environment state s_t, we solve a receding-horizon problem by sampling M noise sequences, rolling out H-step imagined trajectories through f_θ under the current policy, and forming the Monte Carlo objective $\widehat{J}_t(\psi)$ given by discounted predicted rewards plus a terminal value from Q_ϕ. We update ψ for E inner steps using gradients through the full computation graph, then execute the resulting action $a_t = \pi_\psi(s_t)$ in the real environment and repeat at the next state. We report mean and standard deviation of normalized score over multiple evaluation seeds.

Task Name	TD3+BC	IQL	CQL	SAC-RND	ReBRAC	MPCwDWM
halfcheetah-random	30.9 ± 0.4	19.5 ± 0.8	31.1 ± 3.5	27.6 ± 2.1	29.5 ± 1.5	**29.9** ± 1.50
halfcheetah-medium	54.7 ± 0.9	50.0 ± 0.2	46.9 ± 0.4	66.4 ± 1.4	65.6 ± 1.0	**70.05** ± 1.8
halfcheetah-expert	93.4 ± 0.4	95.5 ± 2.1	97.3 ± 1.1	102.6 ± 4.2	105.9 ± 1.7	**106.14** ± 2.1
halfcheetah-medium-expert	89.1 ± 5.6	92.7 ± 2.8	95.0 ± 1.4	**108.1** ± 1.5	101.1 ± 5.2	105.35 ± 1.8
halfcheetah-medium-replay	45.0 ± 1.1	42.1 ± 3.6	45.3 ± 0.3	51.2 ± 3.2	51.0 ± 0.8	**59.89** ± 1.2
halfcheetah-full-replay	75.0 ± 2.5	75.0 ± 0.7	76.9 ± 0.9	81.2 ± 1.3	82.1 ± 1.1	**85.149** ± 0.9
hopper-random	8.5 ± 0.7	10.1 ± 5.9	5.3 ± 0.6	19.6 ± 12.4	8.1 ± 2.4	**8.4**0 ± 1.210
hopper-medium	60.9 ± 7.6	65.2 ± 4.2	61.9 ± 6.4	91.1 ± 10.1	102.0 ± 1.0	**103.38** ± 0.35
hopper-expert	109.6 ± 3.7	108.8 ± 3.1	106.5 ± 9.1	**109.8** ± 0.5	100.1 ± 8.3	104.30 ± 7.03
hopper-medium-expert	87.8 ± 10.5	85.5 ± 29.7	96.9 ± 15.1	109.8 ± 0.6	107.0 ± 6.4	107.420 ± 5.30
hopper-medium-replay	55.1 ± 31.7	89.6 ± 13.2	86.3 ± 7.3	97.2 ± 9.0	98.1 ± 5.3	**103.14** ± 0.45
hopper-full-replay	97.9 ± 17.5	104.4 ± 10.8	101.9 ± 0.6	107.4 ± 0.8	107.1 ± 0.4	**108.77** ± 0.6434
walker2d-random	2.0 ± 3.6	11.3 ± 7.0	5.1 ± 1.7	18.7 ± 6.9	18.4 ± 4.5	18.4 ± 4.5(⋆)
walker2d-medium	77.7 ± 2.9	80.7 ± 3.4	79.5 ± 3.2	**92.7** ± 1.2	82.5 ± 3.6	88.91 ± 0.6
walker2d-expert	110.0 ± 0.6	96.9 ± 32.3	109.3 ± 0.1	104.5 ± 22.8	112.3 ± 0.2	**116.65** ± 0.41
walker2d-medium-expert	110.4 ± 0.6	112.1 ± 0.5	109.1 ± 0.2	104.6 ± 11.2	111.6 ± 0.3	**115.76** ± 1.04
walker2d-medium-replay	68.0 ± 19.2	75.4 ± 9.3	76.8 ± 10.0	89.4 ± 3.8	77.3 ± 7.9	**95.87** ± 1.19
walker2d-full-replay	90.3 ± 5.4	97.5 ± 1.4	94.2 ± 1.9	105.3 ± 3.2	102.2 ± 1.7	**105.76** ± 2.91
Average	70.3	72.9	73.6	82.6	81.2	**85.18**

Table 1: Average normalized score over the final evaluation and ten unseen training seeds on Gym-MuJoCo tasks. TD3+BC IQL CQL SAC-RND and ReBRAC scores are taken from (Tarasov et al., 2023a). The symbol ± represents the standard deviation across the seeds. (⋆) On walker2d-random we did not see an improvement with inference-time optimization and the performance is that of the pre-trained ReBRAC policy.

Task Name	TD3+BC	IQL	CQL	SAC-RND	ReBRAC	MPCwDWM
antmaze-umaze	66.3 ± 6.2	83.3 ± 4.5	74.0	97.0 ± 1.5	97.8 ± 1.0	**98.3** ± 1.1
antmaze-umaze-diverse	53.8 ± 8.5	70.6 ± 3.7	84.0	66.0 ± 25.0	88.3 ± 13.0	88.3 ± 13.0(⋆)
antmaze-medium-play	26.5 ± 18.4	64.6 ± 4.9	61.2	38.5 ± 29.4	84.0 ± 4.2	**86.33** ± 4.8
antmaze-medium-diverse	25.9 ± 15.3	61.7 ± 6.1	53.7	74.7 ± 10.7	76.3 ± 13.5	**84.7** ± 6.7
antmaze-large-play	0.0 ± 0.0	42.5 ± 6.5	15.8	43.9 ± 29.2	60.4 ± 26.1	**70.0** ± 18.2
antmaze-large-diverse	0.0 ± 0.0	27.6 ± 7.8	14.9	45.7 ± 28.5	54.4 ± 25.1	**62.4** ± 26.61
Average	28.7	58.3	50.6	60.9	76.8	**81.77**

Table 2: Average normalized score over the final evaluation and ten unseen training seeds on AntMaze tasks. TD3+BC IQL CQL SAC-RND and ReBRACCQL scores awere taken from (Tarasov et al., 2023a). The symbol ± represents the standard deviation across the seeds. (⋆) On antmaze-umaze-diverse we did not see an improvement with inference-time optimization and the performance is that of the pre-trained ReBRAC policy.

Results. Across D4RL continuous-control benchmarks, MPCwDWM is competitive with and often improves over strong offline RL baselines. It achieves average normalized scores of 85.18 on Gym MuJoCo (vs. SAC-RND 82.6, ReBRAC 81.2) and 81.77 on AntMaze (vs. ReBRAC 76.8), with the largest gains on hard AntMaze variants (large-play 60.4 → 70.0, large-diverse 54.4 → 62.4).

6 Conclusion

We introduced an inference-time adaptation framework for offline RL inspired by model predictive control, in which a pre-trained policy is refined online using the current state and a learned differentiable world model. Our key design is a Diffusion World Model pipeline that supports end-to-end gradient computation through imagined rollouts, enabling direct optimization of policy parameters at inference time using a finite-horizon surrogate objective built from predicted rewards and a terminal value function. Empirically, this inference-time optimization yields consistent gains over strong offline RL baselines across D4RL continuous-control benchmarks, including Gym MuJoCo tasks and AntMaze.

Future work includes reducing inference time via multi step generative world models that predict finite horizon futures in a few passes. Flow matching or mean flow objectives may further cut the cost of differentiating through imagined rollouts. Decoupled variants inspired by Flow Q Learning could enable inference time improvement without backpropagation through time. Finally, offline to online extensions with periodic world model updates may improve robustness under distribution shift.

REFERENCES

A. Ajay, Y. Du, A. Gupta, J. B. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2211.15657.

A. Argenson and G. Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations (ICLR)*, 2021. doi: 10.48550/arXiv.2008.05556. URL https://openreview.net/forum?id=OMNB1G5xzd4.

E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, London, 2 edition, 2007. doi: 10.1007/978-0-85729-398-5.

L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Z. Ding, A. Zhang, Y. Tian, and Q. Zheng. Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning. *arXiv preprint arXiv:2402.03570*, 2024.

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851, 2020.

M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 1273–1286, 2021a. doi: 10.48550/arXiv.2106.02039. URL https://papers.neurips.cc/paper_files/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html.

M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 1273–1286, 2021b. doi: 10.48550/arXiv.2106.02039. URL https://papers.neurips.cc/paper_files/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html.

M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022a.

M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022b.

D. Ki, J. Oh, S.-W. Shim, and B.-J. Lee. Prior-guided diffusion planning for offline reinforcement learning. *arXiv preprint arXiv:2505.10881*, 2025.

R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. doi: 10.48550/arXiv.1906.00949. URL https://papers.nips.cc/paper/2019/hash/2f4d86d9b4d1d7c0b8b6d81e2a9c3f2b-Abstract.html.

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*. Springer, 2012.

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2023.

A. Nair, M. Dalal, A. Gupta, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. In *International Conference on Learning Representations (ICLR)*, 2021. doi: 10.48550/arXiv.2006.09359. URL https://openreview.net/forum?id=OJiM1R3jAtZ.

A. Nikulin, V. Kurenkov, D. Tarasov, and S. Kolesnikov. Anti-exploration by random network distillation. *arXiv preprint arXiv:2301.13616*, 2023. URL https://arxiv.org/abs/2301.13616.

J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, Madison, WI, 2 edition, 2017. 2nd edition. Available at: https://sites.engineering.ucsb.edu/~jbraw/mpc/.

J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2256–2265. PMLR, 2015.

D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2305.09836*, 2023a. doi: 10.48550/arXiv.2305.09836.

D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. doi: 10.48550/arXiv.1911.11361.

T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

T. Yun, S. Yun, J. Lee, and J. Park. Guided trajectory generation with diffusion models for offline model-based optimization. *arXiv preprint arXiv:2407.01624*, 2024.

A Related Works

Offline reinforcement learning. Offline reinforcement learning studies how to learn a high return policy from a fixed dataset while mitigating distribution shift between the learned policy and the logged behavior Lange et al. (2012); Levine et al. (2020). A large family of methods constrains policy improvement to remain close to actions supported by the dataset, either by explicitly regularizing the learned policy toward behavior cloning or by limiting the space of candidate actions used for value based updates Wu et al. (2019); Fujimoto et al. (2019); Kumar et al. (2019); Fujimoto and Gu (2021); Tarasov et al. (2023b). Another family focuses on conservative value estimation to reduce overestimation when evaluating actions that are unlikely under the dataset, with CQL penalizing high values assigned to low density actions Kumar et al. (2020). Complementary approaches avoid explicit out of distribution action maximization during policy improvement by extracting policies from value functions using implicit or advantage weighted updates, as in IQL and AWAC Kostrikov et al. (2022); Nair et al. (2021). Across these lines, the shared objective is to maximize return while ensuring that learning and improvement remain grounded in the support of the offline data.

Generative models for offline reinforcement learning. Recent work replaces simple unimodal parametric policies with conditional generative models that can represent multimodal behaviors present in offline datasets. Sequence modeling approaches treat decision making as conditional generation over trajectories, using return or goal conditioning to produce actions, including Decision Transformer and trajectory modeling formulations Chen et al. (2021); Janner et al. (2021a). Diffusion based methods generate trajectories or action sequences via iterative denoising, enabling flexible behavior synthesis and planning style control, with Diffuser as a representative approach and related developments connected to continuous time generative modeling such as flow matching Janner et al. (2022a); Lipman et al. (2023). In parallel, model based offline reinforcement learning uses learned dynamics to enable short horizon planning or to augment training with synthetic rollouts while controlling model error, including MOPO, MOReL, COMBO, and MBOP Yu et al. (2020); Kidambi et al. (2020); Yu et al. (2021); Argenson and Dulac-Arnold (2021). Diffusion world models extend the dynamics modeling component by learning future state distributions beyond step by step rollout Ding et al. (2024), and several works use generative planners at inference time to sample candidate futures with conditioning or guidance mechanisms, including conditional generative decision making and guided diffusion trajectory generation Ajay et al. (2023); Ki et al. (2025); Yun et al. (2024). These generative and model based directions typically select actions from sampled plans or use learned models to improve training, whereas the focus here is on adapting policy parameters at inference time by backpropagating through a differentiable world model over imagined rollouts.

B Proof of Theorem 4.1

We fix a time t, horizon H, and a noise sequence $\varepsilon_{t:t+H-1}$. All quantities below are defined along the imagined rollout (2). Since the noise is fixed, the rollout is deterministic, hence the gradient $\nabla_\psi L(\psi; \varepsilon_{t:t+H-1})$ is obtained by repeated application of the chain rule.

B.1 Sensitivity recursions

Recall the rollout recursion (2):

$$\tilde{s}_0 = s_t, \qquad \tilde{a}_h = \pi_\psi(\tilde{s}_h), \qquad \tilde{s}_{h+1} = f_\theta(\tilde{s}_h, \tilde{a}_h, \varepsilon_{t+h}), \qquad h = 0, \dots, H-1. \tag{5}$$

Define the state and action sensitivities

$$G_h := \nabla_\psi \tilde{s}_h, \qquad D_h := \nabla_\psi \tilde{a}_h. \tag{6}$$

Since $\tilde{s}_0 = s_t$ does not depend on ψ, we have

$$G_0 = 0. \tag{7}$$

For each h, the action is $\tilde{a}_h = \pi_\psi(\tilde{s}_h)$. By the chain rule,

$$D_h = \nabla_\psi \pi_\psi(\tilde{s}_h) + \nabla_s \pi_\psi(s)\big|_{s=\tilde{s}_h} \nabla_\psi \tilde{s}_h. \tag{8}$$

Introduce the policy Jacobians (as in the theorem statement)

$$\Pi_s(h) := \nabla_s \pi_\psi(s)\big|_{s=\tilde{s}_h}, \qquad \Pi_\psi(h) := \nabla_\psi \pi_\psi(\tilde{s}_h). \tag{9}$$

Substituting (9) into (8) yields

$$D_h = \Pi_s(h)\, G_h + \Pi_\psi(h). \tag{10}$$

For each $h \in \{0, \dots, H-1\}$, the next state is $\tilde{s}_{h+1} = f_\theta(\tilde{s}_h, \tilde{a}_h, \varepsilon_{t+h})$. Differentiating with respect to ψ and applying the chain rule gives

$$G_{h+1} = \nabla_s f_\theta(s, a, \varepsilon_{t+h})\big|_{s=\tilde{s}_h, a=\tilde{a}_h}\, G_h + \nabla_a f_\theta(s, a, \varepsilon_{t+h})\big|_{s=\tilde{s}_h, a=\tilde{a}_h}\, D_h. \tag{11}$$

Introduce the world-model Jacobians (as in the theorem statement)

$$F_s(h) := \nabla_s f_\theta(s, a, \varepsilon_{t+h})\big|_{s=\tilde{s}_h, a=\tilde{a}_h}, \qquad F_a(h) := \nabla_a f_\theta(s, a, \varepsilon_{t+h})\big|_{s=\tilde{s}_h, a=\tilde{a}_h}. \tag{12}$$

Substituting (12) into (11) yields

$$G_{h+1} = F_s(h)\, G_h + F_a(h)\, D_h. \tag{13}$$

Equations (7), (10), and (13) are exactly the sensitivity recursions stated in Theorem 4.1.

B.2 Gradient of the per-noise return

Recall the per-noise return definition (3):

$$L(\psi; \varepsilon_{t:t+H-1}) = \sum_{h=0}^{H-1} \gamma^h\, r_\xi(\tilde{s}_h, \tilde{a}_h) + \gamma^H\, Q_\phi\Big(\tilde{s}_H, \pi_\psi(\tilde{s}_H)\Big). \tag{14}$$

Fix $h \in \{0, \dots, H-1\}$ and define the stage reward $r_h := r_\xi(\tilde{s}_h, \tilde{a}_h)$. By the chain rule,

$$\nabla_\psi r_h = \nabla_s r_\xi(\tilde{s}_h, \tilde{a}_h)\, G_h + \nabla_a r_\xi(\tilde{s}_h, \tilde{a}_h)\, D_h. \tag{15}$$

Introduce the shorthand

$$r_s(h) := \nabla_s r_\xi(\tilde{s}_h, \tilde{a}_h), \qquad r_a(h) := \nabla_a r_\xi(\tilde{s}_h, \tilde{a}_h). \tag{16}$$

Then (15) becomes

$$\nabla_\psi r_\xi(\tilde{s}_h, \tilde{a}_h) = r_s(h)\, G_h + r_a(h)\, D_h. \tag{17}$$

Define the terminal action $\tilde{a}_H := \pi_\psi(\tilde{s}_H)$ and the terminal value $V_T := Q_\phi(\tilde{s}_H, \tilde{a}_H)$. By the chain rule,

$$\nabla_\psi V_T = \nabla_s Q_\phi(\tilde{s}_H, \tilde{a}_H)\, G_H + \nabla_a Q_\phi(\tilde{s}_H, \tilde{a}_H)\, D_H. \tag{18}$$

Introduce the terminal derivatives

$$Q_s := \nabla_s Q_\phi(\tilde{s}_H, \tilde{a}_H), \qquad Q_a := \nabla_a Q_\phi(\tilde{s}_H, \tilde{a}_H). \tag{19}$$

Then (18) becomes

$$\nabla_\psi Q_\phi(\tilde{s}_H, \tilde{a}_H) = Q_s\, G_H + Q_a\, D_H. \tag{20}$$

Here D_H is defined exactly as in (6), and can be expanded via the same policy sensitivity identity (10) with $h = H$:

$$D_H = \Pi_s(H)\, G_H + \Pi_\psi(H). \tag{21}$$

Differentiating (14) and applying (17) and (20) gives

$$\begin{aligned} \nabla_\psi L(\psi; \varepsilon_{t:t+H-1}) &= \sum_{h=0}^{H-1} \gamma^h\, \nabla_\psi r_\xi(\tilde{s}_h, \tilde{a}_h) + \gamma^H\, \nabla_\psi Q_\phi(\tilde{s}_H, \tilde{a}_H) \\ &= \sum_{h=0}^{H-1} \gamma^h \Big(r_s(h)\, G_h + r_a(h)\, D_h\Big) + \gamma^H \Big(Q_s\, G_H + Q_a\, D_H\Big), \end{aligned} \tag{22}$$

which is the gradient expression stated in Theorem 4.1.

Next we derive and prove the Jacobian recursions for $\nabla_a f_\theta(s,a,\varepsilon)$ and $\nabla_s f_\theta(s,a,\varepsilon)$ under the reverse diffusion implementation (1).

Fix (s,a) and a noise pack $\varepsilon = (z_K, z_{K-1}, \dots, z_0)$. Let $\{s^{(k)}\}_{k=0}^K$ be generated by

$$s^{(K)} = g_K(z_K), \tag{23}$$

$$s^{(k-1)} = h_k\big(s^{(k)}, s, a, z_{k-1}\big), \qquad k = K, \dots, 1, \tag{24}$$

$$f_\theta(s,a,\varepsilon) = s^{(0)}. \tag{25}$$

Assume g_K is independent of (s,a) and each $h_k(u,s,a,z)$ is differentiable in (u,s,a).

Define the diffusion-depth sensitivities

$$A_k := \nabla_a s^{(k)} \in \mathbb{R}^{d\times m}, \qquad B_k := \nabla_s s^{(k)} \in \mathbb{R}^{d\times d}. \tag{26}$$

Since $s^{(K)} = g_K(z_K)$ and g_K does not depend on (s,a),

$$A_K = 0, \qquad B_K = 0. \tag{27}$$

Fix $k \in \{1, \dots, K\}$ and write (24) as

$$s^{(k-1)} = h_k(u,s,a,z_{k-1})\Big|_{u=s^{(k)}}. \tag{28}$$

Differentiate (28) with respect to a and apply the chain rule:

$$A_{k-1} = \nabla_a s^{(k-1)} = \frac{\partial h_k}{\partial u}\,\nabla_a s^{(k)} + \frac{\partial h_k}{\partial a} = \frac{\partial h_k}{\partial u}\,A_k + \frac{\partial h_k}{\partial a}, \tag{29}$$

where the partial derivatives are evaluated at

$$(u,s,a,z) = \big(s^{(k)},\ s,\ a,\ z_{k-1}\big). \tag{30}$$

Differentiating (28) with respect to s gives

$$B_{k-1} = \nabla_s s^{(k-1)} = \frac{\partial h_k}{\partial u}\,\nabla_s s^{(k)} + \frac{\partial h_k}{\partial s} = \frac{\partial h_k}{\partial u}\,B_k + \frac{\partial h_k}{\partial s}, \tag{31}$$

with the same evaluation point (30).

Iterating to obtain $\nabla_a f_\theta$ and $\nabla_s f_\theta$. Starting from (27) and iterating (29)–(31) for $k = K, K-1, \dots, 1$ yields A_0 and B_0. Since $f_\theta(s,a,\varepsilon) = s^{(0)}$ by (25),

$$\nabla_a f_\theta(s,a,\varepsilon) = \nabla_a s^{(0)} = A_0, \qquad \nabla_s f_\theta(s,a,\varepsilon) = \nabla_s s^{(0)} = B_0. \tag{32}$$

C Additional Experimental Details

We report the predictive performance of the offline trained diffusion dynamics model and the offline trained reward model across D4RL MuJoCo datasets. Table 3 summarizes reward model accuracy via reward prediction RMSE, while Table 4 reports diffusion model accuracy via MC one step RMSE, with uncertainty reported as standard error and additionally normalized by the number of transitions for the MC one step metric. Overall, both models achieve low error on most datasets, with larger errors on the more challenging expert style settings, most notably `walker2d-expert`.

Task	One step RMSE (± SE / #transitions)
halfcheetah-random	0.2318 ± 0.000010
halfcheetah-medium	0.5494 ± 0.000014
halfcheetah-expert	1.3065 ± 0.000026
halfcheetah-medium-expert	0.5972 ± 0.000013
halfcheetah-medium-replay	0.3996 ± 0.000014
halfcheetah-full-replay	0.4739 ± 0.000014
hopper-random	0.1268 ± 0.004497
hopper-medium	0.0784 ± 0.001281
hopper-expert	0.1375 ± 0.000755
hopper-medium-expert	0.1678 ± 0.000710
hopper-medium-replay	0.1008 ± 0.000764
hopper-full-replay	0.1178 ± 0.002514
walker2d-random	0.6591 ± 0.009463
walker2d-medium	1.4068 ± 0.005905
walker2d-expert	3.3412 ± 0.014321
walker2d-medium-expert	1.5566 ± 0.026789
walker2d-medium-replay	1.1100 ± 0.002611
walker2d-full-replay	1.2560 ± 0.009531
Average	0.7565 ± 0.000031

Table 3: One step RMSE (mean ± standard error) on D4RL MuJoCo datasets.

Task	Reward RMSE (± SE)
halfcheetah-random	0.0477 ± 0.0016
halfcheetah-medium	0.1292 ± 0.0034
halfcheetah-expert	0.2763 ± 0.0076
halfcheetah-medium-expert	0.1622 ± 0.0045
halfcheetah-medium-replay	0.0918 ± 0.0027
halfcheetah-full-replay	0.1164 ± 0.0033
hopper-random	0.0082 ± 0.0003
hopper-medium	0.0720 ± 0.0030
hopper-expert	0.1415 ± 0.0058
hopper-medium-expert	0.0789 ± 0.0033
hopper-medium-replay	0.0192 ± 0.0005
hopper-full-replay	0.0208 ± 0.0005
walker2d-random	0.0244 ± 0.0008
walker2d-medium	0.0779 ± 0.0020
walker2d-expert	0.4902 ± 0.0123
walker2d-medium-expert	0.1060 ± 0.0031
walker2d-medium-replay	0.0459 ± 0.0013
walker2d-full-replay	0.0578 ± 0.0022
Average	0.1092 ± 0.0271

Table 4: Reward prediction RMSE (mean ± standard error) on D4RL MuJoCo datasets.

C.1 Comparison with Generative model based approaches to Offline RL

We compare against representative offline RL and sequence modeling baselines. Conservative Q Learning (CQL) learns a conservative action value function by regularizing values so that out of dataset actions are assigned lower values, reducing overestimation under distribution shift Kumar et al. (2020). Implicit Q Learning (IQL) avoids querying values for unseen actions by fitting a value function with expectile regression and updating the policy with advantage weighted regression

Kostrikov et al. (2022). Decision Transformer (DT) casts offline RL as return conditioned sequence modeling and predicts actions with a Transformer conditioned on the desired return and history Chen et al. (2021). Trajectory Transformer (TT) models trajectories as sequences and plans by decoding high reward trajectories under the learned sequence model Janner et al. (2021b). MOPO learns a dynamics model from offline data and optimizes a policy in the learned model while penalizing rewards using model uncertainty to discourage exploiting model errors Yu et al. (2020). MOReL constructs a pessimistic surrogate MDP that transitions to an absorbing state in regions where the model is unreliable, then learns a policy within this conservative MDP Kidambi et al. (2020). MBOP performs model predictive planning in a learned dynamics model while constraining candidate action sequences with a behavior prior and extending the horizon with a value estimate Argenson and Dulac-Arnold (2021). Finally, Diffuser learns a diffusion model over trajectories and plans by iterative denoising with flexible conditioning, so sampling corresponds to generating high reward behavior consistent with the constraints Janner et al. (2022a).

Table 5 shows that MPCwDWM consistently outperforms prior generative model based offline RL approaches across all three dataset types. In particular, MPCwDWM achieves the best average score and improves substantially over diffusion based trajectory generation (Diffuser) on Medium and Medium Replay, where long horizon compounding errors are most pronounced. It also matches or exceeds the strongest model based baselines on Medium Expert, indicating that inference time policy refinement through a differentiable world model can deliver gains even when offline data quality is high. Overall, these results highlight the benefit of optimizing the policy at test time using imagined rollouts, rather than relying only on sampled plans or purely offline learned policies.

Dataset	Environment	DT	TT	MOPO	MOReL	MBOP	Diffuser	MPCwDWM
Medium Expert	HalfCheetah	86.8	95.0	63.3	53.3	**105.9**	88.9 ± 0.3	105.35 ± 1.8
	Hopper	107.6	**110.0**	23.7	108.7	55.1	103.3 ± 1.3	107.420 ± 5.30
	Walker2d	108.1	101.9	44.6	95.6	70.2	106.9 ± 0.2	**115.76** ± 1.04
Medium	HalfCheetah	42.6	46.9	42.3	42.1	44.6	42.8 ± 0.3	**70.05** ± 1.8
	Hopper	67.6	61.1	28.0	95.4	48.8	74.3 ± 1.4	**103.38** ± 0.35
	Walker2d	74.0	79.0	17.8	77.8	41.0	79.6 ± 0.55	**88.91** ± 0.6
Medium Replay	HalfCheetah	36.6	41.9	53.1	40.2	42.3	37.7 ± 0.5	**59.89** ± 1.2
	Hopper	82.7	91.5	67.5	93.6	12.4	93.6 ± 0.4	**103.14** ± 0.45
	Walker2d	66.6	82.6	39.0	49.8	9.7	70.6 ± 1.6	**95.87** ± 1.19
Average		74.7	78.9	42.1	72.9	47.8	77.5	**94.4**

Table 5: Normalized scores on D4RL MuJoCo tasks. DT, TT, MOPO, MOReL, MBOP, and Diffuser values are taken from Janner et al. (2022b). Bold indicates the best score within each row.