Dynamics-Aligned Diffusion Planning for Offline RL: A Unified Framework with Forward and Inverse Guidance

Anonymous authors
Paper under double-blind review

Abstract

Diffusion-based planning has emerged as a powerful paradigm for offline reinforcement learning (RL). However, existing approaches often overlook the physical constraints imposed by real-world dynamics, resulting in dynamics inconsistency—a mismatch between diffusion-generated trajectories and those feasible under true environment transitions. To address this issue, we propose Dynamics-Aligned Diffusion Planning (DADP), a unified framework that explicitly enforces dynamics consistency during the diffusion denoising process. DADP offers two complementary variants: DADP-F (Forward), which employs a forward dynamics model to ensure state-level feasibility, and DADP-I (Inverse), which leverages an inverse dynamics model to enhance action-level executability. Both variants share a unified guidance formulation that integrates task return optimization and dynamics alignment through gradient-based updates. Experiments on D4RL Maze2D and MuJoCo benchmarks demonstrate that DADP-F and DADP-I outperform state-of-the-art offline RL baselines, effectively reducing dynamics inconsistency and improving long-horizon robustness. This unifies diffusion-based planning with physically grounded dynamics modeling.

1 Introduction

Reinforcement learning (RL) (Sutton, 2018; Silver et al., 2017) has demonstrated remarkable potential in solving complex sequential decision-making problems. However, its application in critical domains such as medical development (Fatemi et al., 2022), autonomous driving (Fang et al., 2022), and robotics control (Sinha et al., 2022) is significantly constrained by the impracticality of its trial-and-error mechanism. This limitation has spurred the development of offline RL (Kumar et al., 2020; Jiang et al., 2023; Yu et al., 2020), which focuses on learning effective policies directly from pre-collected datasets.

Because offline RL relies solely on fixed, behavior-driven datasets, most algorithms adopt conservative objectives that favor in-distribution actions. While such strategies mitigate extrapolation errors, they also induce myopic policy optimization (Jiang et al., 2023), focusing excessively on short-term rewards while neglecting the long-term consequences of early decisions. This limited foresight makes it challenging to synthesize reliable multi-step action sequences—an ability essential for modern robotic and autonomous systems. The problem is further exacerbated by perception noise, actuation uncertainty, and distributional shifts between the behavior policy and the learned policy, which lead to error accumulation and performance degradation over extended horizons.

Recent advances in diffusion-based planning provide a new probabilistic perspective on long-horizon decision-making by framing it as optimal trajectory distribution inference. Inspired by the control-as-inference framework (Levine, 2018) and recent progress in diffusion generative models (Croitoru et al., 2023; Janner et al., 2022), these methods train diffusion models to transform Gaussian noise into complete trajectories through iterative denoising. Rather than learning a policy to generate trajectories sequentially, they operate directly in the trajectory space: diffusion models are trained to map noise sampled from a simple prior to full trajectories via a learned denoising process. By estimating a posterior distribution over optimal trajectories, these frameworks can subsequently guide the extraction or refinement of effective policies. This formulation

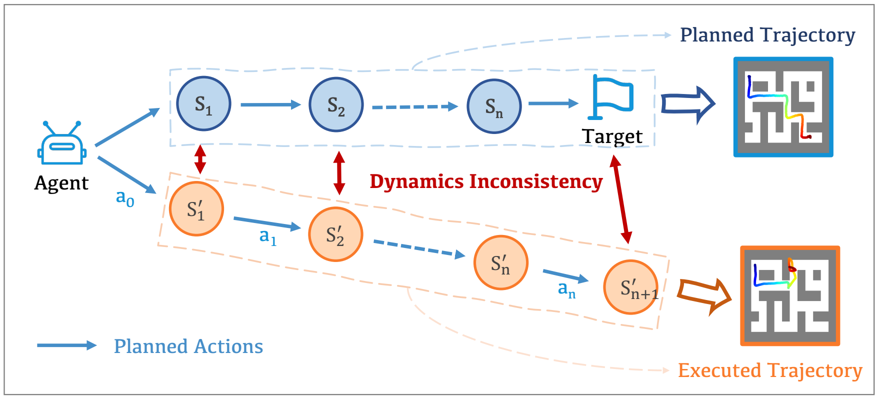


Figure 1: Illustration of dynamics inconsistency in diffusion-based planning. The diffusion model generates a planned trajectory (blue) toward the target through denoising in trajectory space. When executed in the real environment, the actual trajectory (orange) deviates due to dynamics inconsistency—the mismatch between predicted and true transitions. This phenomenon motivates the proposed Dynamics-Aligned Diffusion Planning (DADP), which enforces forward (state-level) or inverse (action-level) consistency during trajectory generation.

captures long-range contextual dependencies, enabling the generation of trajectories that are both robust to epistemic uncertainty and well-suited for challenging offline RL scenarios.

Despite recent advances in diffusion-based planning, existing approaches still struggle to capture the fine-grained local dynamics of real environments. As a result, the trajectories they generate may appear globally coherent but become physically infeasible upon execution, violating the true transition constraints of the system. We refer to this phenomenon as dynamics inconsistency—a fundamental limitation of current diffusion-based offline RL methods. Dynamics inconsistency arises when the predicted next state or action deviates from the actual environment transition, leading to cumulative errors that gradually manifest as a divergence between planned and executed trajectories over long horizons (Figure 1). Such inconsistencies severely undermine the reliability and robustness of diffusion-based control, particularly in complex continuous domains.

To address this challenge, we propose Dynamics-Aligned Diffusion Planning (DADP), a unified framework that explicitly enforces alignment between diffusion-generated trajectories and environment dynamics. DADP augments the standard denoising process with a dynamics-alignment energy term that serves as a differentiable physics-consistency constraint during reverse diffusion. At each denoising step, the model jointly optimizes task reward and local dynamic fidelity, producing trajectories that are not only high-reward but also physically realizable and dynamically stable. Conceptually, DADP transforms diffusion-based planning from pure generative inference into physics-guided optimization, bridging the gap between probabilistic modeling and real-world feasibility. Moreover, this integration enables the planner to capture richer temporal correlations and more faithful motion continuity, providing a principled way to couple learned dynamics with trajectory generation under a unified probabilistic framework.

To comprehensively address dynamics inconsistency, we further decompose it into two complementary aspects and propose two independent variants under a unified framework. DADP-F (Forward) incorporates a forward dynamics model to penalize physically invalid state transitions, ensuring that generated next states are smoothly reachable given the current states and actions within the learned dynamics manifold. In contrast, DADP-I (Inverse) employs an inverse dynamics model to correct infeasible or non-executable actions during denoising, guaranteeing that generated controls are both consistent and realizable for achieving the desired state transitions. Each variant independently enforces dynamics alignment from different perspectives: DADP-F focuses on maintaining valid and physically plausible state transitions, while DADP-I emphasizes the executability and stability of generated actions. Together, they provide complementary insights into improving physical consistency, robustness, and long-horizon stability in diffusion-based planning.

Our main contributions are summarized as follows:

1. **Unified Framework.** We propose Dynamics-Aligned Diffusion Planning (DADP), a unified framework that explicitly integrates environment dynamics into diffusion-based trajectory generation for offline reinforcement learning.
2. **Dual Implementations.** We develop two independent variants—DADP-F (Forward) and DADP-I (Inverse)—which enforce state-level and action-level consistency, respectively, addressing dynamics alignment from predictive and corrective perspectives.
3. **Theoretical Insight.** We provide an analysis showing that explicit dynamics alignment mitigates compounding errors and stabilizes open-loop diffusion planning.
4. **Comprehensive Evaluation.** Extensive experiments on D4RL Maze2D and MuJoCo benchmarks demonstrate that both variants consistently outperform state-of-the-art diffusion-based and model-based offline RL methods in terms of planning accuracy and robustness.

2 Related Works

Offline Reinforcement Learning. Offline reinforcement learning (Offline RL) aims to learn high-performing policies from static datasets while maintaining conservatism to mitigate distributional shift. Most approaches achieve this by imposing value or policy constraints that keep the learned policy within or close to the behavior distribution. Early value-based methods such as Batch-Constrained Deep Q-learning (BCQ) (Fujimoto et al., 2019) regularize the Q-target using behavior-clipped actions. Conservative Q-Learning (CQL) (Kumar et al., 2020) and Implicit Q-Learning (IQL) (Kostrikov et al., 2021) further constrain the learned value function through advantage-weighted or expectile regression objectives, effectively avoiding out-of-distribution (OOD) actions. Although these methods alleviate distributional bias, they often exhibit myopic optimization tendencies, focusing on immediate rewards and struggling to produce reliable long-horizon behaviors in complex control tasks.

Model-Based Reinforcement Learning (MBRL). Model-based reinforcement learning improves sample efficiency by explicitly learning environment dynamics for planning or policy optimization. Representative works such as MOPO (Yu et al., 2020) and MOReL (Kidambi et al., 2020) mitigate model bias by penalizing uncertainty or terminating rollouts that deviate from the data distribution. COMBO (Yu et al., 2021) combines model-based and value-based regularization to enhance offline policy learning. Despite their benefits, these approaches fundamentally depend on model accuracy: even small prediction errors can accumulate over time, resulting in unstable rollouts and degraded performance in high-dimensional continuous environments.

Diffusion-Based Planning for Reinforcement Learning. Recent advances employ diffusion models to directly generate trajectories in offline reinforcement learning. Diffuser (Janner et al., 2022) introduced a denoising diffusion probabilistic model conditioned on returns to synthesize entire state–action sequences. Building on this idea, Decision Diffuser (Ajay et al., 2022) incorporated conditional guidance for desired outcomes, while hierarchical extensions such as HDMI (Li et al., 2023) enabled multi-level trajectory planning. More recently, adaptive diffusion-based planning (Zhou et al., 2024) has introduced uncertainty estimation for online re-planning. While these methods effectively capture global trajectory structures, they often overlook local physical consistency, resulting in trajectories that may be statistically valid yet dynamically infeasible when executed. Our proposed Dynamics-Aligned Diffusion Planning (DADP) explicitly addresses this issue by integrating forward and inverse dynamics constraints into the denoising process, ensuring that generated trajectories remain both high-reward and physically realizable.

3 Method

3.1 Problem Formulation

Offline reinforcement learning (Offline RL) aims to learn optimal decision-making policies purely from pre-collected datasets, without any further interaction with the environment. The environment is modeled as a

Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, P is the transition kernel, r the reward function, γ the discount factor, and ρ_0 the initial state distribution. An agent follows a policy $\pi : \mathcal{S} \to \mathcal{A}$ to generate trajectories $\tau = \{(s_t, a_t)\}_{t=0}^{T-1}$ of horizon T. The optimal policy can be interpreted as producing the first action a_0 according to an optimal trajectory distribution:

$$\pi^*(a|s) = p^*(\tau|s_0 = s)(a_0), \tag{1}$$

where $p^*(\tau|s_0 = s)$ denotes the distribution over optimal trajectories starting from the initial state s.

In the offline RL setting, the agent has access only to a static dataset $\mathcal{D}$ with empirical trajectory distribution $P_{\mathcal{D}}(\tau)$. The learning objective becomes to infer an optimal trajectory distribution from this fixed dataset, which can be formulated as:

$$\hat{p}^*(\tau) = \arg\max_{\hat{p}} \mathbb{E}_{\tau\sim\hat{p}(\tau)}[\log P_{\mathcal{D}}(\tau)] + \mathbb{E}_{\tau\sim\hat{p}(\tau)}[\mathcal{J}(\tau)], \tag{2}$$

where $\mathcal{J}(\tau) = \sum_{t=0}^{T-1} r(s_t, a_t)$ denotes the cumulative return.

This dual-objective formulation jointly encourages distributional alignment with the offline dataset and reward maximization for optimal control. The likelihood term serves as an implicit regularizer, alleviating sparse or delayed reward issues that often arise in long-horizon planning. This formulation naturally lends itself to diffusion-based generative modeling, where the goal is to approximate the optimal trajectory distribution through iterative denoising.

3.2 Diffusion-based Trajectory Modeling

Diffusion probabilistic models (Sohl-Dickstein et al., 2015) are powerful generative frameworks that synthesize complex data by iteratively denoising samples through a Markovian process. In the trajectory generation setting, a diffusion model learns to map Gaussian noise to complete trajectories via a reverse diffusion process. Specifically, the generative process is defined as a sequence of reverse transitions $p(\tau^{i-1}|\tau^i)$ that invert a forward diffusion process $q(\tau^i|\tau^{i-1})$, where the latter progressively corrupts clean trajectories by adding Gaussian noise at each step. The marginal data distribution modeled by the diffusion process can be written as:

$$p(\tau^0) = \int p(\tau^N) \prod_{i=1}^{N} p(\tau^{i-1}|\tau^i)\, d\tau^{1:N}, \tag{3}$$

where $p(\tau^N)$ is a standard Gaussian prior and τ^0 denotes the clean trajectory. The reverse transitions are typically parameterized as Gaussian distributions:

$$p(\tau^{i-1}|\tau^i) = \mathcal{N}(\tau^{i-1} \mid \mu(\tau^i, i), \Sigma(i)), \tag{4}$$

where $\mu(\tau^i, i)$ and $\Sigma(i)$ represent the mean and covariance of the reverse process, learned by a neural network conditioned on the noisy trajectory τ^i and the diffusion step i. The forward process $q(\tau^i|\tau^{i-1})$ is defined as a fixed Gaussian corruption kernel (Sohl-Dickstein et al., 2015).

To integrate diffusion-based generation into decision-making, we adopt the control-as-inference perspective (Levine, 2018), which formulates optimal control as a probabilistic inference problem. Let $\mathcal{O}_t$ denote a binary optimality variable at time t, where $p(\mathcal{O}_t = 1) = \exp(r(s_t, a_t))$. The optimal trajectory distribution can then be expressed as a posterior:

$$p^*(\tau) = p(\tau \mid \mathcal{O}_{0:T} = 1) \propto p(\tau)\, p(\mathcal{O}_{0:T} = 1|\tau). \tag{5}$$

This formulation encourages sampling trajectories that are both behaviorally plausible (high $p(\tau)$) and reward-optimal (high $p(\mathcal{O}_{0:T} = 1|\tau)$).

Following (Janner et al., 2022), when the optimality likelihood $p(\mathcal{O}_{0:T} = 1|\tau)$ is smooth, the reverse diffusion transitions can be approximated by:

$$p(\tau^{i-1}|\tau^i, \mathcal{O}_{0:T}) \approx \mathcal{N}\big(\tau^{i-1}; \mu + \Sigma\nabla_\mu\mathcal{J}(\mu), \Sigma\big), \tag{6}$$

where μ, Σ are parameters of the unconditional diffusion model, and $\nabla_\mu \mathcal{J}(\mu)$ represents a reward-guided gradient that steers denoising toward higher-return trajectories. This perspective bridges generative modeling and reward optimization, forming the basis for our proposed dynamics-aligned diffusion framework introduced in the next section.

3.3 Quantifying Dynamics Inconsistency

While diffusion-based planning effectively models global trajectory distributions, it often fails to capture fine-grained local transition dynamics, resulting in discrepancies between generated and physically realizable trajectories. To quantify these discrepancies, we define a dynamics inconsistency metric that measures how much the diffusion-generated transitions deviate from those governed by the true environment dynamics.

Formally, dynamics inconsistency can be assessed from two complementary perspectives:

State-level inconsistency (Forward). This measures the deviation between the next state s_{t+1} predicted by the diffusion model and the next state $\hat{s}_{t+1} = M_f(s_t, a_t)$ predicted by a learned forward dynamics model:

$$E_T^{\mathrm{F}} = \sum_{t=0}^{T-1} \|s_{t+1} - \hat{s}_{t+1}\|_2. \tag{7}$$

It reflects how much the generated state transition violates physical feasibility.

Action-level inconsistency (Inverse). This measures the deviation between the generated action a_t and the corresponding action $\hat{a}_t = M_i(s_t, s_{t+1})$ inferred by a learned inverse dynamics model:

$$E_T^{\mathrm{I}} = \sum_{t=0}^{T-1} \|a_t - \hat{a}_t\|_2. \tag{8}$$

It captures how inconsistent the generated action is with that physically required to realize the same transition.

Both metrics serve as differentiable objectives that quantify local violations of environment dynamics. E_T^{F} constrains the set of reachable states, whereas E_T^{I} constrains the executability of actions. These two complementary objectives are integrated into the DADP framework, where DADP-F enforces forward (state-level) consistency and DADP-I enforces inverse (action-level) consistency, together providing bidirectional regularization of dynamics during diffusion-based trajectory generation.

3.4 Learning the Dynamics Models

To enable differentiable and uncertainty-aware consistency guidance, both the forward and inverse dynamics models in our framework are implemented as variational autoencoders (VAEs). A probabilistic latent-variable formulation offers two main advantages over deterministic regression: (1) it captures uncertainty, enabling robustness under stochastic or partially observed transitions; and (2) it provides a smooth, differentiable latent space that supports gradient-based consistency guidance during diffusion denoising. This formulation mitigates compounding prediction errors common in deterministic models and improves generalization to unseen state–action pairs.

Formally, given observed transitions (s_t, a_t, s_{t+1}), the encoder maps inputs to a latent representation z_ϕ, while the decoder reconstructs the conditional distribution of the target variable—either the next state or the action—depending on the model type. Assuming a standard Gaussian prior $p(z) = \mathcal{N}(0, I)$, both models are trained by maximizing the evidence lower bound (ELBO):

$$\mathcal{L}_{\mathrm{ELBO}} = \mathbb{E}_{q_\phi(z|x)}[\log p_\psi(y|z)] - \mathrm{KL}(q_\phi(z|x) \,\|\, p(z)), \tag{9}$$

where (x, y) denotes the input–target pair and ϕ, ψ are the parameters of the encoder and decoder, respectively. We instantiate this structure in two complementary forms:

Forward dynamics model (M_f). The forward dynamics model predicts the next state given the current state and action, approximating the true transition function $P(s_{t+1}|s_t, a_t)$:

$$M_f(s_t, a_t) = \mathcal{N}(\mu_f(z_\phi(s_t, a_t)),\ \Sigma_f(z_\phi(s_t, a_t))). \tag{10}$$

It is trained to minimize the mean-squared error between predicted and observed states:

$$\mathcal{L}_f = \sum_{t=0}^{T-1} \|M_f(s_t, a_t) - s_{t+1}\|_2^2. \tag{11}$$

This model enforces predictive consistency, ensuring that denoised trajectories evolve toward physically reachable states.

Inverse dynamics model (M_i). The inverse dynamics model estimates the action that induces the transition between two consecutive states, modeling $P(a_t|s_t, s_{t+1})$:

$$M_i(s_t, s_{t+1}) = \mathcal{N}(\mu_i(z_\phi(s_t, s_{t+1})),\ \Sigma_i(z_\phi(s_t, s_{t+1}))). \tag{12}$$

It learns a latent action manifold that captures multiple feasible controls for the same transition, providing robust feasibility correction during denoising.

In practice, the forward and inverse dynamics models are used separately to support two DADP variants. DADP-F employs the forward model to guide trajectory generation toward state-level consistency, while DADP-I utilizes the inverse model to enforce action-level consistency. Each model acts as a differentiable prior that regularizes the diffusion process, ensuring that generated trajectories remain consistent with the underlying environment dynamics.

3.5 Dynamics-Aligned Diffusion Planning (DADP)

The proposed Dynamics-Aligned Diffusion Planning (DADP) framework integrates learned dynamics models into the reverse diffusion process to mitigate dynamics inconsistency. The key idea is to introduce a dynamics-alignment constraint during denoising, ensuring that generated trajectories remain both reward-optimal and physically feasible throughout the diffusion process.

During planning, two pretrained models are employed: a diffusion model $p_\theta(\tau)$ and a dynamics model M, which can be either the forward model M_f or the inverse model M_i, depending on the DADP variant. Trajectory sampling begins by initializing τ^N from a standard Gaussian prior $\mathcal{N}(0, I)$, with the initial state fixed to the agent's current observation. The reverse diffusion process iteratively denoises the trajectory over N steps, where each step is represented by $p_\theta(\tau^{i-1}|\tau^i)$. At each iteration, the diffusion model outputs a mean $\mu_\theta(\tau^i)$ and covariance Σ^i that govern the sampling distribution, ensuring both stability and diversity during denoising.

To enforce local dynamics alignment, we compute a dynamics inconsistency score at each iteration. Depending on the variant, the formulation is as follows:

State-level alignment (DADP-F).

$$E^F(\tau) = \sum_{t=0}^{T-1} \gamma_t \|M_f(s_t, a_t) - s_{t+1}\|^2, \tag{13}$$

Action-level alignment (DADP-I).

$$E^I(\tau) = \sum_{t=0}^{T-1} \gamma_t \|M_i(s_t, s_{t+1}) - a_t\|^2, \tag{14}$$

where $\gamma_t \in [0, 1]$ denotes a timestep-dependent weight that balances contributions across the trajectory or accounts for uncertainty in the dynamics model.

At each denoising step, the trajectory is refined using gradient-based guidance that balances task rewards and dynamics consistency. The gradient of the augmented objective is computed as

$$g = \nabla_\mu(\lambda \mathcal{J} - \beta E), \tag{15}$$

where $\mathcal{J}$ represents the cumulative reward defined in Eq. 2, λ controls the reward weighting, and β determines the strength of dynamics alignment. The next trajectory is then sampled as

$$\tau^{i-1} \sim \mathcal{N}(\boldsymbol{\mu} + \alpha \Sigma g, \Sigma^i), \tag{16}$$

where α adjusts the influence of the guidance term while Σ^i preserves sampling variance for exploration.

To ensure task-specific feasibility, we apply constraints $\mathcal{C}$ during denoising:

$$\mathcal{C}(\tau) = \{s_0 = s_{\text{current}},\ s_T = s_{\text{goal}} \text{ (if specified)}\}, \tag{17}$$

where s_{current} is the current state and s_{goal} is an optional target. These constraints ensure that generated trajectories start from valid states and, when applicable, terminate at goal positions.

In summary, DADP extends standard diffusion-based planning by incorporating explicit dynamics alignment. The forward variant (DADP-F) promotes predictive reachability through state-level consistency, while the inverse variant (DADP-I) enforces action-level feasibility. Both independently enhance the physical validity and stability of long-horizon diffusion planning. The overall procedure is summarized in Algorithm 1.

Algorithm 1 Dynamics-Aligned Diffusion Planning (DADP) for Offline Trajectory Generation

```
Input: Diffusion model $\boldsymbol{\mu}_\theta$; Dynamics model $\mathbf{M}$ (either forward $M_f$ or inverse $M_i$); Planning horizon $T$; Number of diffusion steps $N$; Scaling factors $\alpha$, $\lambda$, $\beta$; Covariances $\{\Sigma^i\}_{i=1}^N$; Constraints $\mathcal{C}$.
Initialize noisy trajectory $\tau^N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; observe initial state $s_{\text{current}}$.
for $i = N, \dots, 1$ do
    Compute denoised mean $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_\theta(\tau^i)$.
    Extract state–action pairs $\{(s_t, a_t, s_{t+1})\}_{t=0}^{T-1}$ from $\boldsymbol{\mu}$.
    Compute cumulative reward $\mathcal{J} \leftarrow \sum_{t=0}^{T-1} r(s_t, a_t)$.
    Compute dynamics inconsistency term:
        $E \leftarrow \begin{cases} \sum_{t=0}^{T-1} \|M_f(s_t, a_t) - s_{t+1}\|^2, & \text{for DADP-F} \\ \sum_{t=0}^{T-1} \|M_i(s_t, s_{t+1}) - a_t\|^2, & \text{for DADP-I} \end{cases}$
    Compute joint guidance gradient: $g \leftarrow \nabla_\mu(\lambda \mathcal{J} - \beta E)$.
    Sample denoised trajectory: $\tau^{i-1} \sim \mathcal{N}(\boldsymbol{\mu} + \alpha \Sigma^i g, \Sigma^i)$.
    Project $\tau^{i-1}$ to satisfy constraints $\mathcal{C}$ (e.g., $s_0 = s_{\text{current}}$, optional $s_T = s_{\text{goal}}$).
end for
Output: Final planned trajectory $\tau^0$.
```

4 Experiments

We conduct extensive experiments to comprehensively evaluate the proposed Dynamics-Aligned Diffusion Planning (DADP) framework. The study is designed to address the following key research questions:

1. Can DADP effectively reduce dynamics inconsistency and mitigate error accumulation in long-horizon planning?

2. Do the two variants, DADP-F and DADP-I, enhance robustness and generalization across diverse continuous-control environments?

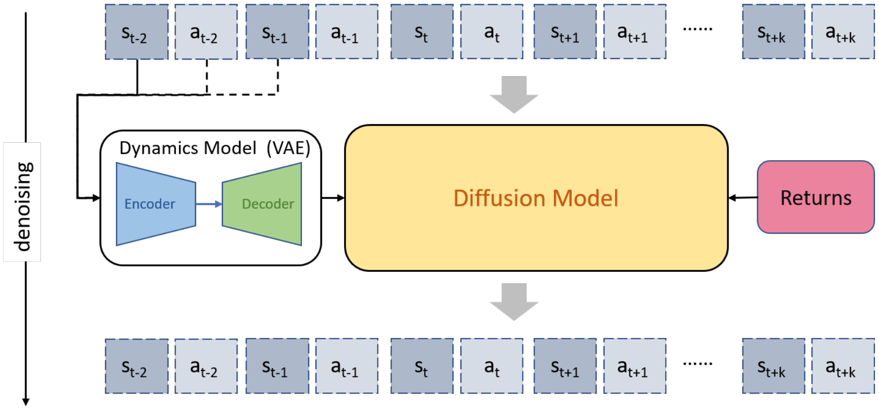


Figure 2: Overview of the proposed Dynamics-Aligned Diffusion Planning (DADP) framework. The dynamics model, implemented as a variational autoencoder (VAE), provides local physical consistency guidance during the diffusion-based trajectory denoising process. For the forward variant (DADP-F), the dynamics model predicts the next state given (s_t, a_t), while for the inverse variant (DADP-I), it infers the corresponding action from (s_t, s_{t+1}). The diffusion model then refines the trajectory by jointly optimizing for reward maximization and dynamics alignment.

We evaluate our approach on two widely used D4RL benchmarks: Maze2D for long-horizon, sparse-reward navigation, and MuJoCo for continuous robotic control. Our implementation builds upon the Diffuser codebase, employing a U-Net backbone for trajectory denoising. Both forward and inverse dynamics models are implemented as variational autoencoders (VAEs) with fully connected encoder–decoder architectures. All models are trained using the Adam optimizer, with uniform weighting ($\gamma_t = 1$). Hyperparameters α, λ, and β are tuned per environment via validation to ensure stable convergence.

In the following sections, we first describe the experimental setup and baseline configurations, then present quantitative results on Maze2D and MuJoCo benchmarks, followed by an in-depth analysis of the effects of dynamics alignment in DADP-F and DADP-I.

4.1 Maze2D: Evaluating Long-Horizon Planning

We evaluate the proposed DADP framework on the D4RL Maze2D benchmark, a suite of long-horizon navigation tasks characterized by sparse rewards. Maze2D comprises three configurations—umaze, medium, and large—corresponding to increasing maze complexity and planning horizons. The agent receives a reward of 1 only upon reaching the goal, so higher cumulative returns indicate shorter completion times within the fixed horizon. Following standard evaluation protocols, we report normalized scores (reward × 100), where higher values reflect more efficient planning and better task performance.

Baselines. We compare DADP against representative model-free and model-based offline reinforcement learning algorithms. Model-free baselines include Batch-Constrained Deep Q-Learning (BCQ) (Fujimoto et al., 2019), Conservative Q-Learning (CQL) (Kumar et al., 2020), and Implicit Q-Learning (IQL) (Kostrikov et al., 2021). Model-based methods include ROMI (Wang et al., 2021), the original Diffuser (Janner et al., 2022), and improved variants such as Diffuser-HER, Diffuser-AM (Kim et al., 2024), and Hierarchical Diffusion for Offline Decision Making (HDMI) (Li et al., 2023). Together, these baselines cover diverse paradigms of value regularization, model-based rollout, and diffusion-based planning.

Results. Table 1 presents normalized performance scores on both Maze2D and Multi2D benchmarks. DADP-F and DADP-I consistently outperform all baselines across maze sizes and task settings. In the single-task scenario, DADP-F achieves an average normalized score of 137.8, surpassing Diffuser (119.5) and HDMI (123.5). DADP-I attains a similar score of 137.5, demonstrating higher robustness across random seeds. In the more challenging Maze2D-Large environment, DADP-F achieves 147.7—representing a 20% improvement over Diffuser (123.0)—highlighting the benefits of explicit dynamics alignment.

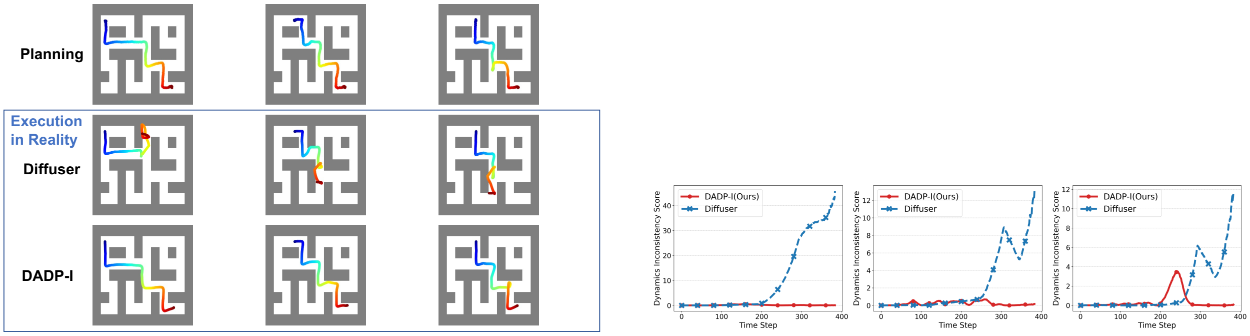
In the multi-task setting, where start and goal positions are randomized, DADP-F and DADP-I again yield the best overall performance, achieving averages of 144.2 and 145.9, respectively. Both variants generalize effectively to unseen configurations, while traditional diffusion-based planners show degraded performance under distributional shift.

Representative planning and execution results in Maze2D-Large are shown in Figure 3a. The first row shows planned trajectories, while the second and third rows correspond to executions under Diffuser and DADP-I, respectively. Each column depicts an independent episode. Diffuser often deviates between planned and executed paths due to cumulative modeling errors, whereas DADP-I maintains close alignment, confirming the stability benefits of dynamics alignment.

Figure 3b further provides a quantitative comparison of dynamics inconsistency across representative episodes. Diffuser exhibits rapidly increasing inconsistency over time, indicating compounding errors, while DADP-I maintains low deviation throughout execution. These findings confirm that inverse dynamics guidance effectively suppresses error propagation and improves physical consistency in long-horizon planning.

Table 1: Normalized scores on D4RL Maze2D and Multi2D benchmarks.

Environment	**BCQ**	**CQL**	**IQL**	**ROMI**	**DD**	**HDMI**	**Diffuser**	**Diffuser (HER)**	**Diffuser (AM)**	**DADP-F (Ours)**	**DADP-I (Ours)**
Maze2D U-Maze	12.8	5.7	47.4	**139.5**	116.2	120.1	113.9	125.4	121.4	130.2 ± 0.8	131.5 ± 0.7
Maze2D Medium	8.3	5.0	34.9	82.4	122.3	121.8	121.5	130.3	127.2	**135.4** ± 0.5	**136.8** ± 0.8
Maze2D Large	6.2	12.5	58.6	83.1	125.9	128.6	123.0	135.8	135.0	**147.7** ± 3.6	**144.2** ± 4.1
Single-task Average	9.1	7.7	47.0	101.7	121.5	123.5	119.5	130.5	127.9	**137.8**	**137.5**
Multi2D U-Maze	-	-	24.8	-	128.2	131.3	128.9	132.7	135.4	**136.8** ± 1.3	**137.0** ± 1.6
Multi2D Medium	-	-	12.1	-	129.7	131.6	127.2	133.0	137.8	**139.7** ± 0.7	**142.1** ± 0.9
Multi2D Large	-	-	13.9	-	130.5	135.4	132.1	139.2	145.7	**156.1** ± 2.2	**158.7** ± 2.5
Multi-task Average	-	-	16.9	-	129.5	132.8	129.4	135.0	139.6	**144.2**	**145.9**



(a) Visualization of planned and executed trajectories on the Maze2D-Large benchmark. Each column corresponds to an independent episode. The first row shows trajectories planned by the Diffuser, while the second and third rows depict their actual executions under Diffuser and DADP-I, respectively. Compared to Diffuser, DADP-I maintains closer alignment between planned and executed paths, indicating improved dynamics consistency.

(b) Dynamics inconsistency scores for the three episodes illustrated in (a). Each plot shows the deviation between planned and executed trajectories for Diffuser and DADP-I. Lower scores indicate higher fidelity and better execution consistency.

Figure 3: Comparison between trajectory visualization and dynamics consistency analysis on the Maze2D-Large benchmark. (a) visualizes planned and executed trajectories, while (b) reports corresponding dynamics inconsistency scores.

4.2 MuJoCo: Robust Continuous Control

Datasets. We further evaluate DADP on the D4RL MuJoCo continuous-control benchmarks, including HalfCheetah, Hopper, and Walker2d. Each environment is evaluated under three datasets—medium-expert, medium, and medium-replay—that capture increasing degrees of data imperfection and distributional shift.

Each dataset contains one million transitions generated by policies of different quality levels. The medium-expert dataset combines trajectories from expert and medium policies in equal proportions, the medium dataset is derived from a partially trained Soft Actor–Critic (SAC) policy, and the medium-replay dataset aggregates all replay buffer transitions up to a medium-performance policy.

Baselines. We compare DADP with representative offline reinforcement learning algorithms from both model-free and model-based paradigms. Model-free baselines include Behavior Cloning (BC), Conservative Q-Learning (CQL) (Kumar et al., 2020), Implicit Q-Learning (IQL) (Kostrikov et al., 2021), and Decision Transformer (DT) (Chen et al., 2021). For model-based approaches, we consider MOPO (Yu et al., 2020) and Diffuser (Janner et al., 2022). This comprehensive set of baselines enables a fair comparison across policy-learning, model-based, and diffusion-based paradigms, isolating the contribution of dynamics alignment in improving robustness and stability.

Results. Table 2 reports normalized scores across all MuJoCo environments and datasets. Both DADP-F and DADP-I show superior performance compared to most baseline methods, where DADP-I attains the highest average normalized score of 81.2 across all evaluated tasks. DADP-F excels on tasks with smoother and more predictable dynamics, such as HalfCheetah and Walker2d, whereas DADP-I demonstrates superior robustness in stochastic environments like Hopper. Compared to Diffuser, both variants achieve substantial gains—particularly on the medium and medium-replay datasets—indicating that explicit dynamics alignment effectively mitigates model bias and stabilizes long-horizon control. Overall, these results confirm that incorporating forward or inverse dynamics guidance significantly enhances the stability, robustness, and generalization of diffusion-based offline reinforcement learning.

Table 2: Performance comparison on the D4RL MuJoCo benchmark. Each entry reports normalized scores for different offline RL algorithms across three datasets (medium-expert, medium, and medium-replay). Results for DADP-F and DADP-I are averaged over 100 random seeds and reported as mean ± standard error. The highest score for each task is highlighted in bold.

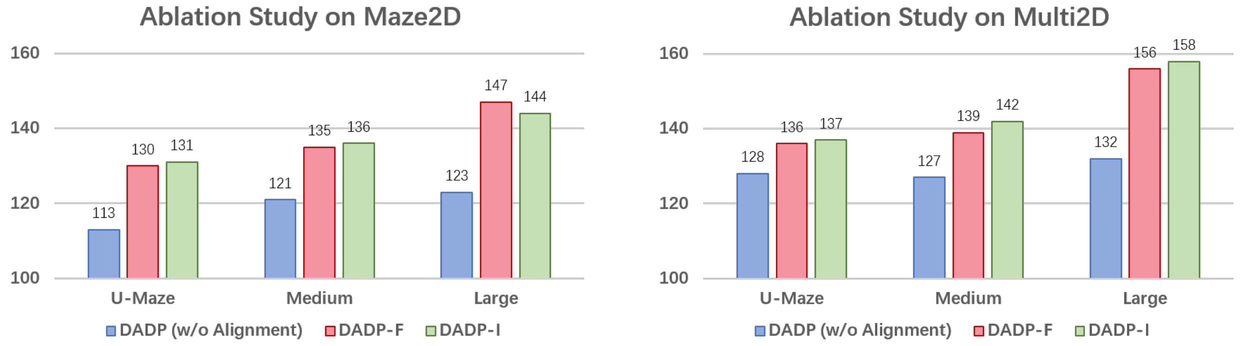
Dataset	Environment	CQL	IQL	DT	MOPO	Diffuser	DADP-F (Ours)	DADP-I (Ours)
Medium-Expert	HalfCheetah	91.6	86.7	86.8	63.3	88.9	89.6 ± 0.1	**92.3 ± 0.1**
Medium-Expert	Hopper	105.4	91.5	107.6	23.7	103.3	104.4 ± 1.0	**108.1 ± 1.0**
Medium-Expert	Walker2d	108.8	**109.6**	108.1	44.6	106.9	107.6 ± 0.1	108.0 ± 0.1
Medium	HalfCheetah	44.0	**47.4**	42.6	42.3	42.8	44.9 ± 0.1	43.9 ± 0.1
Medium	Hopper	58.5	66.3	67.6	28.0	74.3	77.3 ± 0.7	**96.2 ± 0.7**
Medium	Walker2d	72.5	78.3	74.0	17.8	79.6	**81.8 ± 0.4**	80.5 ± 0.4
Medium-Replay	HalfCheetah	45.5	44.2	36.6	**53.1**	37.7	38.7 ± 0.3	38.8 ± 0.3
Medium-Replay	Hopper	95.0	94.7	82.7	67.5	93.6	95.2 ± 0.4	**95.6 ± 0.5**
Medium-Replay	Walker2d	77.2	73.9	66.6	39.0	70.6	**77.9 ± 2.4**	71.9 ± 1.8
Average		77.6	77.0	74.7	42.1	77.5	**79.7**	**81.2**

4.3 Ablation Study and Sensitivity Analysis

To assess the contribution of the dynamics-alignment mechanism, we conduct ablation experiments on both the Maze2D and MuJoCo benchmarks by removing the dynamics-consistency term from DADP. As shown in Figure 4, excluding this term (DADP w/o Alignment) leads to a consistent drop in normalized scores across all Maze2D variants under both single-task and multi-task settings. This confirms that the alignment component effectively mitigates long-horizon error accumulation and enforces physical feasibility during trajectory generation.

We further extend the ablation to the MuJoCo control tasks (Table 3), where both DADP-F and DADP-I outperform the unaligned variant across all datasets. DADP-F performs better in environments with smoother dynamics such as HalfCheetah and Walker2d, while DADP-I exhibits higher robustness in stochastic and underactuated environments such as Hopper. These results demonstrate that dynamics alignment provides consistent benefits across diverse control regimes and varying environment complexities.

To analyze the robustness of DADP with respect to hyperparameter choices, we perform sensitivity studies on the inverse-aligned variant (DADP-I) using the MuJoCo benchmark. Here, λ controls the weighting of the reward guidance term, and β adjusts the strength of the dynamics-alignment constraint during denoising. The step-size coefficient α is found to be relatively insensitive and is fixed to 1 in all experiments. Figure 5 presents the sensitivity curves for λ (top) and β (bottom) across three environments: HalfCheetah, Hopper, and Walker2d. Performance remains stable over several orders of magnitude for both parameters, highlighting that DADP-I is robust to hyperparameter variations and requires minimal tuning effort.



(a)

(b)

Figure 4: Ablation study results on the Maze2D and Multi2D benchmarks. (a) Maze2D represents single-task planning with fixed start and goal locations, while (b) Multi2D evaluates the multi-task setting where both start and goal positions are randomly sampled across episodes. Each bar shows the normalized score of DADP variants with and without dynamics alignment.

Table 3: Ablation study results on the D4RL MuJoCo benchmark. Each dataset group (Medium-Expert, Medium, Medium-Replay) contains three continuous control environments: HalfCheetah, Hopper, and Walker2d.

Method	**Medium-Expert**			**Medium**			**Medium-Replay**		
	HalfCheetah	Hopper	Walker2d	HalfCheetah	Hopper	Walker2d	HalfCheetah	Hopper	Walker2d
DADP (w/o Alignment)	88.9	103.3	106.9	42.8	74.3	79.6	37.7	93.6	70.6
DADP-F	89.6	104.4	107.6	44.9	77.3	81.8	38.7	95.2	77.9
DADP-I	92.3	108.1	108.0	43.9	96.2	80.5	38.8	95.6	71.9

4.4 Open-Loop Multi-Step Execution Analysis

Most previous diffusion-based planners adopt a closed-loop execution scheme, where the agent replans after every action using the most recent observation. While this strategy effectively limits error accumulation, it incurs substantial computational overhead. In practical scenarios, however, reducing planning frequency by executing multiple actions per plan—an open-loop strategy—can greatly improve efficiency.

Figure 6 illustrates the trade-off between average rollout time and performance for DADP-I on the MuJoCo Hopper-medium task. As the number of executed actions per plan increases, the overall computation time decreases substantially, with only marginal degradation in normalized performance.

These results demonstrate that DADP-I maintains strong robustness under reduced planning frequency, achieving an effective balance between computational efficiency and control performance—an essential property for real-time deployment of diffusion-based planners.

5 Limitations and Future Work

Despite its promising results, DADP still faces several limitations. First, the training and inference of diffusion models are computationally expensive, limiting scalability to high-dimensional environments or real-time applications. Second, the framework relies on pretrained dynamics models, whose accuracy directly

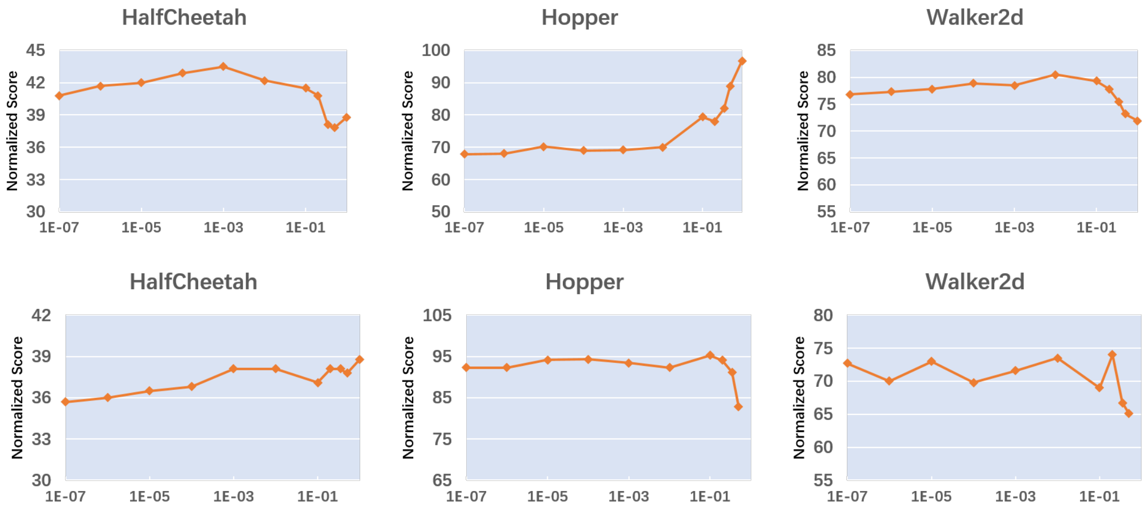


Figure 5: Sensitivity analysis of hyperparameters λ and β on the MuJoCo benchmark. The top figure reports results on the Medium setting, showing the effect of λ, which controls the weighting of the reward objective. The bottom figure shows the Medium-Replay setting, illustrating the influence of β, the dynamics alignment weight. Both λ and β are plotted on a logarithmic scale.

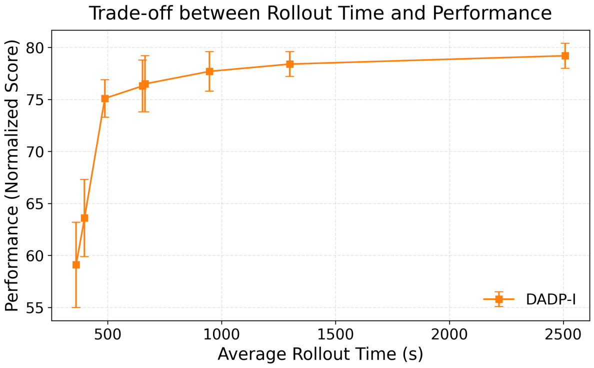


Figure 6: Trade-off between rollout time and performance on the MuJoCo Hopper-medium task. Each point represents the average performance of DADP-I under different open-loop execution lengths. The results show that DADP-I maintains stable performance as rollout time decreases, achieving an effective balance between computational efficiency and control accuracy.

affects performance—potentially introducing model bias under severe distribution shifts. Finally, DADP assumes static offline datasets, without online adaptation to non-stationary environments. Future work will explore more efficient sampling strategies to accelerate diffusion inference, as well as adaptive dynamics modeling techniques that enable online fine-tuning and continual learning in evolving real-world scenarios.

6 Conclusion

This paper presented Dynamics-Aligned Diffusion Planning (DADP), a unified framework that integrates explicit dynamics modeling into diffusion-based trajectory generation to address dynamics inconsistency in offline reinforcement learning. By incorporating forward and inverse dynamics guidance, DADP enforces physical feasibility during denoising and enhances stability in long-horizon planning. Extensive experiments on the D4RL Maze2D and MuJoCo benchmarks demonstrate that DADP significantly improves planning accuracy, robustness, and generalization over prior diffusion-based methods.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023.

Xing Fang, Qichao Zhang, Yinfeng Gao, and Dongbin Zhao. Offline reinforcement learning for autonomous driving with real world driving data. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3417–3422. IEEE, 2022.

Mehdi Fatemi, Mary Wu, Jeremy Petch, Walter Nelson, Stuart J Connolly, Alexander Benz, Anthony Carnicelli, and Marzyeh Ghassemi. Semi-markov offline reinforcement learning for healthcare. In *Conference on Health, Inference, and Learning*, pp. 119–137. PMLR, 2022.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915, 2022.

Ke Jiang, Jia-Yu Yao, and Xiaoyang Tan. Recovering from out-of-sample states via inverse dynamics in offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:38815–38826, 2023.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

Sungyoon Kim, Yunseon Choi, Daiki E Matsunaga, and Kee-Eung Kim. Stitching sub-trajectories with conditional diffusion model for goal-conditioned offline rl. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13160–13167, 2024.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

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

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pp. 20035–20064. PMLR, 2023.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pp. 907–917. PMLR, 2022.

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.

Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

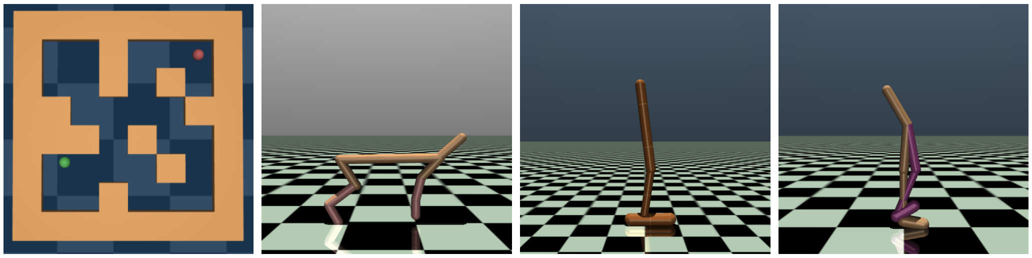
Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Siyuan Zhou, Yilun Du, Shun Zhang, Mengdi Xu, Yikang Shen, Wei Xiao, Dit-Yan Yeung, and Chuang Gan. Adaptive online replanning with diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

A Dataset Details



(a) Maze (b) HalfCheetah (c) Hopper (d) Walker2d

Figure 7: (a) Maze: A point-mass (green dot) agent needs to plan a path from the starting position to reach the target location (red dot). (b) HalfCheetah: A two-dimensional cheetah robot that controls its joints to run as fast as possible to the right. (c) Hopper: A two-dimensional monopod robot where the agent needs to control its hinge to perform a rightward hopping motion. (d) Walker2d: Based on Hopper, an additional leg is added, requiring the agent to coordinate the movements of both legs to achieve fast walking while maintaining balance.

A.1 Maze2D

The Maze2D benchmark in D4RL (Fu et al., 2020) is designed for evaluating long-horizon navigation and sparse-reward planning. It consists of three environments: U-Maze, Medium, and Large, which progressively increase in complexity and trajectory length. Each dataset contains trajectories collected by a goal-conditioned controller navigating through 2D mazes of varying sizes. The agent's state consists of the (x, y) location and velocities. A sparse reward of +1 is provided only upon reaching the goal, while intermediate steps receive zero reward. Following the standard D4RL evaluation protocol, we compute normalized scores by first normalizing the cumulative episode rewards between the minimum and maximum returns of reference (random and expert) policies, and then scaling the result by 100. Thus, a score of 100 corresponds to the expert-level performance, while 0 represents random behavior. The final score is obtained by averaging normalized returns over all evaluation episodes.

A.2 MuJoCo

The MuJoCo benchmark in D4RL contains three standard continuous control tasks: HalfCheetah, Hopper, and Walker2d. Each task is evaluated on three offline datasets—medium-expert, medium, and medium-replay—each containing one million transitions generated by Soft Actor-Critic (SAC) agents trained to different degrees of proficiency.

- **Medium-Expert:** 50% expert and 50% medium-level trajectories mixed uniformly.
- **Medium:** trajectories from a partially trained SAC policy paused at mid-training.
- **Medium-Replay:** all transitions from the replay buffer during SAC training up to the medium performance level.

All environments provide continuous state and action spaces with dense rewards defined by task-specific physical performance metrics (e.g., forward velocity, energy cost, stability). We follow the D4RL normalization protocol, where a score of 0 corresponds to a random policy and 100 to an expert policy.

A.3 Data Usage Protocol

All experiments strictly follow the experimental setup of Diffuser (Janner et al., 2022). Datasets are used exactly as provided by D4RL without any additional filtering, augmentation, or relabeling. We ensure fair comparison by adopting identical training and evaluation splits across all methods. For the Maze2D benchmark, each experiment is averaged over 150 random seeds, while for the MuJoCo benchmark, results are averaged over 100 random seeds. All reported results include the mean and standard error of normalized scores.

B Model and Network Architecture Details

B.1 Diffusion Model

The diffusion model used in DADP follows the same architecture as the original Diffuser (Janner et al., 2022). It adopts a one-dimensional temporal U-Net composed of six repeated residual blocks, designed to capture hierarchical temporal dependencies within trajectory sequences. Each residual block contains two temporal convolution layers, each followed by group normalization (**?**) and a Mish activation function (**?**). Timestep embeddings are generated by a single fully connected layer and are added to the activation of the first temporal convolution within each block to provide diffusion-step awareness. This structure enables stable denoising in the trajectory space while maintaining temporal smoothness and inter-step consistency.

B.2 Return Predictor

The return predictor $J(\tau)$ shares the same backbone as the encoder portion of the U-Net used in the diffusion model, excluding the decoding layers. A final linear projection layer maps the aggregated hidden representation to a scalar value representing the predicted return. This network provides differentiable reward guidance, allowing gradient-based control signals to steer the diffusion denoising process toward high-return trajectories.

B.3 Dynamics Model

The dynamics model, used for both forward and inverse consistency guidance, is implemented as a variational autoencoder (VAE). Each VAE consists of a three-layer fully connected encoder and decoder, each with a hidden dimension of 256 and ReLU activations. The encoder outputs the mean and log standard deviation of a 32-dimensional latent variable z, sampled via the reparameterization trick. The latent variable captures stochasticity and multi-modality in the environment dynamics, enabling smooth and differentiable consistency gradients.

C Training Configuration

C.1 Optimization and Training Setup

All models are trained using the Adam optimizer with default momentum parameters. For the diffusion model, we use a learning rate of 2×10^{-4} and a batch size of 32. For the dynamics models (both forward and inverse), the learning rate is set to 1×10^{-3} with a batch size of 64.

All models are trained using the same datasets as those used for the diffusion process. Each task (Maze2D or MuJoCo) is trained on its corresponding D4RL dataset independently, ensuring that the diffusion and dynamics models are both exposed to consistent trajectory distributions. This alignment between training datasets guarantees that the dynamics priors provide coherent and task-specific consistency guidance during planning.

C.2 Diffusion and Return Predictor Training

The diffusion model is trained following the standard denoising diffusion probabilistic modeling (DDPM) objective:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\tau,\epsilon,i}\left[\|\epsilon - \epsilon_\theta(\tau^i, i)\|_2^2\right], \tag{18}$$

where ϵ_θ denotes the noise prediction network parameterized by the temporal U-Net. We set the diffusion horizon $N = 20$ in Mujoco and $N = 256$ in Maze2d. The return predictor is jointly trained to minimize the mean squared error between predicted and true returns, sharing the encoder weights of the diffusion model for representational consistency.

C.3 Dynamics Model Training

Both the forward and inverse dynamics models are trained independently using the variational autoencoder (VAE) objective. Each model employs a latent dimension of 32, with both encoder and decoder composed of three fully connected layers of size 256, each followed by ReLU activation. The encoder outputs the mean and log standard deviation of the latent variable, which is sampled via the reparameterization trick for differentiability.

Training is conducted using the Adam optimizer with a learning rate of 1×10^{-3} and a batch size of 64. Each model is trained for 3K epochs using the same dataset as the diffusion model, ensuring consistent trajectory distributions across components. Model checkpoints are saved periodically (every 1000 epochs) and the best model is selected based on validation loss. All training runs employ the same data preprocessing and normalization pipeline as in the diffusion model, with no data augmentation or relabeling applied.

C.4 Hyperparameter Settings

During planning, the key guidance coefficients are selected per environment via validation: $\alpha = 1.0$ (step size), $\lambda \in [0.8, 1.0]$ (reward weight), and $\beta \in [10^{-2}, 10^{0}]$ (dynamics alignment weight). These ranges balance reward optimization and dynamics consistency.

C.5 Hardware and Software Configurations

All experiments are conducted on a workstation equipped with two Intel Xeon Gold 5218 CPUs (2.30GHz, 32 cores in total), four NVIDIA GeForce RTX 2080 Ti GPUs (11GB each), and 256GB of system memory. The operating system is Ubuntu Linux with CUDA 13.0 and driver version 580.95.05.