PlanDQ: Hierarchical Plan Orchestration via D-Conductor and Q-Performer

Chang Chen [1] **Junyeob Baek** [2] **Fei Deng** [1] **Kenji Kawaguchi** [3] **Caglar Gulcehre** [4] **Sungjin Ahn** [2]

Abstract

Despite the recent advancements in offline RL, no unified algorithm could achieve superior performance across a broad range of tasks. Offline *value function learning*, in particular, struggles with sparse-reward, long-horizon tasks due to the difficulty of solving credit assignment and extrapolation errors that accumulates as the horizon of the task grows. On the other hand, models that can perform well in long-horizon tasks are designed specifically for goal-conditioned tasks, which commonly perform worse than value function learning methods on short-horizon, dense-reward scenarios. To bridge this gap, we propose a hierarchical planner designed for offline RL called PlanDQ. PlanDQ incorporates a diffusion-based planner at the high level, named D-Conductor, which guides the low-level policy through sub-goals. At the low level, we used a Q-learning based approach called the Q-Performer to accomplish these sub-goals. Our experimental results suggest that PlanDQ can achieve superior or competitive performance on D4RL continuous control benchmark tasks as well as AntMaze, Kitchen, and Calvin as long-horizon tasks.

1. Introduction

Offline reinforcement learning (RL) (Agarwal et al., 2020; Gulcehre et al., 2020) aims to address the challenges of learning policies from a fixed offline dataset without needing additional costly online interactions with the environment. This ability is particularly important in scenarios where real-time interaction is either impractical or risky, such as healthcare, autonomous driving, and robotics (Sinha et al., 2021; De Lima & Krohling, 2021; Tang et al., 2022). However, offline RL presents unique challenges, the most significant of which is the distributional shift between the behavior policy and the learned policy. Additionally, the quality and diversity of the offline data significantly impact the performance and generalizability of the learned policies (Levine et al., 2020; Bhargava et al., 2023).

In response to these challenges, recent advances in policy learning highlighted the benefits of integrating expressive diffusion models into offline RL. Among those works, the pioneering works like Diffuser (Janner et al., 2022a) and subsequent models (Li et al., 2023; Hong et al., 2023b; Chen et al., 2024) focused on modeling trajectory distributions. Thus, planning with these learned models is equivalent to sampling from them, showing notable improvements, especially in domains requiring long-horizon planning. However, these diffusion planners often fall short in tasks characterized by short horizons and dense rewards. In contrast, another line of research, utilizing diffusion models to represent complex behavior policies (Wang et al., 2023; Lu et al., 2023), are optimized for maximizing expected Q values and show superior performance in dense reward settings. Nonetheless, these value-function learning methods often exhibit limitations in long-horizon, sparse-reward scenarios due to the distinct learning paradigms they employ.

As summarized in Table 1, value-based methods update the value function at each timestep independently, propagating optimal future rewards backward. Although effective in short-horizon, dense-reward settings, the value propagation over lengthy time horizons poses inherent challenges (Bhargava et al., 2023), necessitating the hierarchical structure in long-horizon tasks (Park et al., 2023; Li et al., 2023; Hong et al., 2023b; Chen et al., 2024). On the other hand, sequence modeling methods struggle with stitching optimal policies from sub-optimal trajectories in short-horizon tasks with dense rewards, an essential skill for offline RL (Yamagata et al., 2023; Kumar et al., 2022; Fu et al., 2020; Hong et al., 2023a). This poses difficulty in learning an efficient low-level policy due to the short-horizon property of sub-tasks. Moreover, our empirical observations indicate that, without Bellman updating, the guidance function in diffusion-based sequence modeling methods tends to converge to sub-optimal solutions. A more detailed discussion on Diffuser and Q-Learning for short-horizon problems is provided in Appendix B.

[1]Rutgers University [2]KAIST [3]National University of Singapore [4]EPFL. Correspondence to: Chang Chen <chang.chen@rutgers.edu>, Sungjin Ahn <sungjin.ahn@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

This naturally leads to the question: "Is it possible to devise an algorithm that excels in both settings?" Our analysis above suggests an affirmative response. We hypothesize that combining a sequence modeling method at the high level with a low-level value learning policy would be effective in both scenarios. To realize this concept, we introduce PlanDQ, a hierarchical planning framework for offline RL. Specifically, PlanDQ employs a diffusion-based planner, termed D-Conductor, for high-level planning while leveraging a Q-learning method, referred to as Q-Performer, for efficiently tackling low-level sub-tasks.

Our main contributions can be outlined as follows: 1) We propose a novel hierarchical planning method, PlanDQ, for offline decision-making; 2) Through empirical evaluation, PlanDQ is shown to either outperform or match the performance of existing state-of-the-art methods across a variety of tasks, demonstrating its effectiveness in both long-horizon and short-horizon settings. 3) We provide evaluation reports for other possible orchestrating architectures, affirming the necessity and efficiency of PlanDQ. 4) Our empirical investigations further highlight that the guidance function within the Diffuser model is prone to converge towards sub-optimal solutions when applied to noisy data in short-horizon tasks, leading to inferior performance compared to value learning methods.

Table 1. Comparison between offline *value function learning* and planning methods under different scenarios according to our experiments on the continuous control tasks.

Properties	Value function learning	Sequence modeling
Long-horizon Reasoning	✗(Flat policy)	✔
Bellman Optimality	✔(Short-horizon only)	✗

2. Preliminaries

2.1. Offline Reinforcement Learning

This work focuses on the *offline reinforcement learning* (RL) framework. The offline RL, also known as batch RL, is a variant of RL where the learning process is conducted on a fixed static dataset $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)\}$ collected from behavior policies that we may not have access to. The general RL problem can be formulated as a Markov Decision Process (MDP): $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathbf{s}_0\}$, with state space $\mathcal{S}$, action space $\mathcal{A}$, environment dynamics $\mathcal{P}(s'|s,a)$ representing probability of transitioning to state $\mathbf{s}'$ from state $\mathbf{s}$ after taking action $\mathbf{a}$, reward function $\mathcal{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ defining the expected immediate reward, r, after transitioning from state $\mathbf{s}$ to state $\mathbf{s}'$, discount factor $\gamma \in [0, 1]$, and initial state distribution $\mathbf{s}_0$. The goal of offline RL is to learn a policy $\pi_\theta(a|s)$ that maximizes the cumulative discounted reward purely from an offline dataset without any online interactions:

$$\pi = \max_{\pi} \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] . \tag{1}$$

During this learning process, policy evaluation and policy improvement alternate: a parameterized critic network $Q_\phi(\mathbf{s}, \mathbf{a})$ is optimized to minimize the following temporal difference (TD) loss: $\mathcal{L}_{\text{TD}}(\phi) = \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\left(r + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q_{\hat{\phi}}(\mathbf{s}', \mathbf{a}') - Q_\phi(\mathbf{s}, \mathbf{a}) \right)^2 \right]$, where $Q_{\hat{\phi}(\mathbf{s},\mathbf{a})}$, introduced to stabilize the training, is the target network that keeps a lagged copy of the weight of $Q_\phi(\mathbf{s}, \mathbf{a})$. The parameterized policy π_θ is trained to maximize the expected Q value: $\mathcal{L}_\pi = \mathbb{E}_{\mathbf{s}\in\mathcal{D}, \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}[Q_\phi(\mathbf{s}, \mathbf{a})]$.

While promising, offline RL faces significant challenges due to the function approximation errors encountered with out-of-distribution actions. To mitigate these issues, policy regularization is often required to ensure the learned policy does not deviate too far from the behavior policy (Wang et al., 2022).

2.2. Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Model (DDPM)(Ho et al., 2020) is a latent generative model that learns the data distribution $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:M}) \mathrm{d}\mathbf{x}_{1:M}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_M$ are latent variables that share the same dimensionality as the data $\mathbf{x}_0$. After training, DDPM generates samples through an M-step iterative denoising process, starting from a Gaussian noise $\mathbf{x}_M \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$p_\theta(\mathbf{x}_0) = \int p(\mathbf{x}_M) \prod_{m=0}^{M-1} p_\theta(\mathbf{x}_m \mid \mathbf{x}_{m+1}) \, \mathrm{d}\mathbf{x}_{1:M} \tag{2}$$

$$p_\theta(\mathbf{x}_m \mid \mathbf{x}_{m+1}) = \mathcal{N}(\mathbf{x}_m; \boldsymbol{\mu}_\theta(\mathbf{x}_{m+1}), \sigma_m^2 \mathbf{I}) . \tag{3}$$

To learn a DDPM, the approximate posterior of the latents is given by a fixed diffusion process that gradually adds Gaussian noise to the data:

$$q(\mathbf{x}_{1:M}|\mathbf{x}_0) := \prod_{m=1}^{M} q(\mathbf{x}_m|\mathbf{x}_{m-1}) , \tag{4}$$

$$q(\mathbf{x}_m|\mathbf{x}_{m-1}) := \mathcal{N}(\mathbf{x}_m; \sqrt{1-\beta_m}\mathbf{x}_{m-1}, \beta_m \mathbf{I}) , \tag{5}$$

where β_m is a pre-defined variance schedule $\sqrt{1-\beta_m} \to 0$ as $m \to \infty$, ensuring $\mathbf{x}_M$ would be a pure standard Gaussian noise. The training objective is to maximize the evidence lower bound defined as $\mathbb{E}_q \left[\ln \frac{p_\theta(\mathbf{x}_{0:M})}{q(\mathbf{x}_{1:M}|\mathbf{x}_0)} \right]$.

In practice, the learnable mean $\mu_\theta(\mathbf{x}_m)$ is often parameterized as a linear combination of the latent $\mathbf{x}_m$ and the output of a noise-prediction U-net $\boldsymbol{\epsilon}$ (Ronneberger et al., 2015). The training objective is simplified (Ho et al., 2020) as:

$$\mathcal{L}_\theta = \mathbb{E} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_m)\|^2 \right] . \tag{6}$$

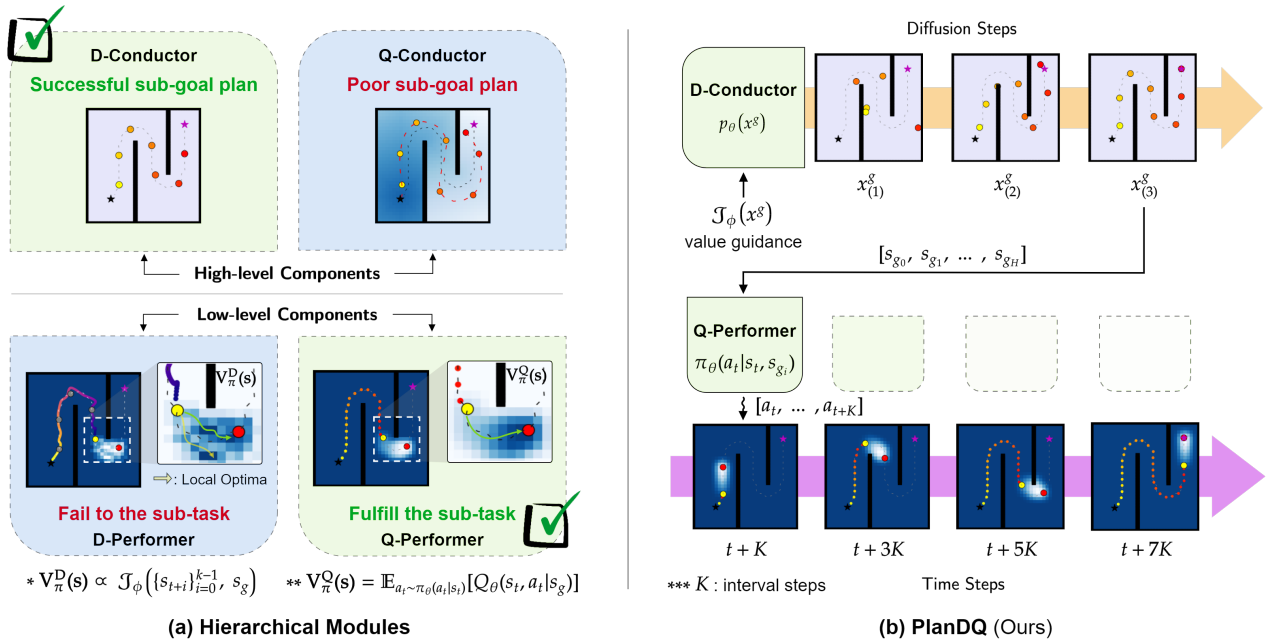


Figure 1. **Overview.** To explore the optimal hierarchical planning architecture, we examine four hierarchical modules, *check* 1-(a), consisting of two conductors for high-level components and two performers for low-level components. And, we introduce a novel hierarchical planning architecture, called PlanDQ, which takes the D-conductor as a high-level component and the Q-performer as a low-level component, *check* 1-(b) The planning process of PlanDQ.

2.3. Diffusion Models for Reinforcement Learning

Current research applying diffusion models into Reinforcement Learning (RL) primarily focuses on three distinct applications: employing diffusion models as planners, as policy functions, and as tools for data synthesis. Given the scope of our work, we will briefly discuss the first two categories: using diffusion models as planners and as policy functions.

Diffusion Planners. Diffuser (Janner et al., 2022a) was first proposed to learn a diffusion-based planner from the offline dataset. The key idea is to re-structure the trajectory of states and actions into a two-dimensional array:

$$\mathbf{x} = \begin{bmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \dots & \mathbf{s}_T \\ \mathbf{a}_0 & \mathbf{a}_1 & \dots & \mathbf{a}_T \end{bmatrix} . \tag{7}$$

A diffusion probabilistic model $p_\theta(\mathbf{x})$ is then trained to model the data distribution. After training, Diffuser can plan a trajectory via sampling $\mathbf{x} \sim p_\theta(\mathbf{x})$. To sample a plan for a specific task, Diffuser separately trains a guidance function $\mathcal{J}_\phi(\mathbf{x})$ to predict the return $R(\mathbf{x})$ of the trajectory $\mathbf{x}$ given a corrupted trajectory $\mathbf{x}_m$ as input.

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{x},m,\epsilon}\left[\|R(\mathbf{x}) - \mathcal{J}_\phi(\mathbf{x}_m)\|^2\right] , \tag{8}$$

Thus, a plan that tries to maximize the expected return can be sampled from the perturbed distribution, $\tilde{p}_\theta(\mathbf{x}) \propto p_\theta(\mathbf{x}) \exp(\mathcal{J}_\phi(\mathbf{x}))$, via classifier-guided sampling:

$$\tilde{\boldsymbol{\mu}} \leftarrow \boldsymbol{\mu}_\theta(\mathbf{x}_{m+1}) + \omega\sigma_m^2 \nabla_{\mathbf{x}_m} \mathcal{J}_\phi(\mathbf{x}_m)|_{\mathbf{x}_m = \boldsymbol{\mu}_\theta(\mathbf{x}_{m+1})}, \tag{9}$$

where the hyperparameter ω controls the gradient scaling.

Follow-up works (Li et al., 2023; Hong et al., 2023a; Chen et al., 2024) have extended Diffuser with hierarchical structures and showed noticeable improvement on long-horizon tasks. The basic idea is to deploy Diffuser as a high-level planner to model a sub-goal sequence and train a low-level policy to follow the sub-goals. The trajectory consisted of sub-goals at the high level is formatted as:

$$\mathbf{x}^g = \begin{bmatrix} \mathbf{s}_{g_0} & \mathbf{s}_{g_1} & \dots & \mathbf{s}_{g_H} \end{bmatrix} . \tag{10}$$

, where H denotes the subgoal planning horizon.

Diffusion Policy Functions Diffusion Q-learning (DQL) (Wang et al., 2022) firstly explores the advantages of modeling the complex behavior policy with a diffusion model: $\pi_\theta(\mathbf{a} \mid \mathbf{s}) = p_\theta(\mathbf{a}_{0:N} \mid \mathbf{s})$. To improve the policy, in addition to the diffusion reconstruction loss, the policy is also trained to maximize the expected return: $\mathcal{L}_\pi(\theta) = \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}_0\sim\pi_\theta}[Q_\phi(\mathbf{s}, \mathbf{a}_0)]$. The end sample, $\mathbf{a}_0$, of the desnoising process $p_\theta(\mathbf{a}_{0:N} \mid \mathbf{s}) = \mathcal{N}(\mathbf{a}_N; \mathbf{0}, \mathbf{I}) \prod_{i=1}^N p_\theta(\mathbf{a}_{i-1} \mid \mathbf{a}_i, \mathbf{s})$ is used for RL evaluation.

3. Method

In this section, we detail the design considerations behind our proposed methodology. Specifically, in Section 3.1, we describe the development of our high-level planner. Sec-

tion 3.2 describes how to implement our goal-conditioned low-level policy. In Section 3.3, we explore alternative hierarchical configurations and potential challenges. The overview structure of PlanDQ can be found in Figure 1.

3.1. D-Conductor

Aligned with our hypothesis, we favor a sequence modeling approach for the high-level planner. Concurrently, our aim for the high-level module is to generate plausible sub-goals or sub-task abstraction (i.e., skills) that guide the low-level policy toward maximizing expected task returns. Various prior studies have explored different methods for implementing this high-level conductor (Sudhakaran & Risi, 2023; Ma et al., 2023; Li et al., 2023; Hong et al., 2023b), within which the Hierarchical Diffuser (HD) proposed by Chen et al., 2024 standing out for its effectiveness and simplicity. To this end, following HD, we deploy diffusion models as our sub-goal planner, termed D-Conductor. Specifically, the sub-goal sequences are formatted with sampled intermediate states with fixed time interval K:

$$\mathbf{x}^g = \begin{bmatrix}\mathbf{s}_{g_0} & \mathbf{s}_{g_1} & \dots & \mathbf{s}_{g_H}\end{bmatrix} = \begin{bmatrix}\mathbf{s}_0 & \mathbf{s}_K & \dots & \mathbf{s}_{HK}\end{bmatrix} . \quad (11)$$

Same as HD or Diffuser, a diffusion model $p_\theta(\mathbf{x}^g)$ is trained to minimize the simplified objective as in equation 6.

While this generally works well for long-horizon tasks with sparse reward, to guide the plan towards maximizing the expected task return in a dense-reward setting, a separate guidance function $\mathcal{J}_\phi(\mathbf{x}^g)$ is required. The gradients of $\mathcal{J}_\phi$ guide the trajectory sampling procedure by modifying the means $\boldsymbol{\mu}$ of the denoising process as in equation 9. Moreover, in this case, including the actions of the intermediate steps within trajectory representation helps the value function learning (Chen et al., 2024). Here, we adopt the same strategy, and as a result, $\mathbf{x}^g$ becomes:

$$\mathbf{x}^g = \begin{bmatrix} \mathbf{s}_0 & \mathbf{s}_K & \dots & \mathbf{s}_{HK} \\ \mathbf{a}_0 & \mathbf{a}_K & \dots & \mathbf{a}_{HK} \\ \mathbf{a}_1 & \mathbf{a}_{K+1} & \dots & \mathbf{a}_{HK+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_{K-1} & \mathbf{a}_{2K-1} & \dots & \mathbf{a}_{(H+1)K-1} \end{bmatrix} . \quad (12)$$

3.2. Q-Performer

Continuing with our hypothesis, we opt for a value function learning method at the low level. This preference is due to the nature of sub-tasks encountered by the low-level policy are typically short horizons and dense rewards (augmented with intrinsic rewards). Value function learning methods can theoretically achieve Bellman optimality through efficient credit assignment. However, directly applying off-policy RL methods to offline learning will suffer from the extrapolation error problem (Fujimoto et al., 2019). Drawing inspiration

Algorithm 1 PlanDQ: D-Conductor Training

1: **Input:** Initialized diffusion network p_θ, value predictor $\mathcal{J}_\phi$, offline dataset D, discount factor γ
2: **for** each iteration **do**
3: Sample a trajectory $\tau = \{(\mathbf{s}_t, r_t)\}^T$
4: Extract state sequence $\{\mathbf{s}_t\}^{HK}$ to form $\mathbf{x}^q$
5: Compute return $R(\tau) = \sum_t^T \gamma^t r_t$
6: Sample $m \sim Uniform(\{1, \dots, M\})$, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
7: Update p_θ by minimizing Equation 6
8: Update $\mathcal{J}_\phi$ by minimizing Equation 9
9: **end for**=0

from recent breakthroughs that utilize expressive diffusion models to capture complex behavior policies, we propose employing a goal-conditioned diffusion model as our low-level policy. Specifically, given the current state $\mathbf{s}$ and sub-goal $\mathbf{s}_g$ from the D-Conductor, we define our Q-performer:

$$\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{s}_g) = p_\theta(\mathbf{a}^M) \prod_{m=1}^{M} p_\theta(\mathbf{a}_{m-1}|\mathbf{a}_m, \mathbf{s}, \mathbf{s}_g) . \quad (13)$$

The objective for policy regularization can be written by:

$$\mathcal{L}_d(\theta) = \mathbb{E}_{m,\epsilon,\mathbf{a}^0,\mathbf{s},\mathbf{s}_g} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta^Q}(\mathbf{a}^m, m; \mathbf{s}, \mathbf{s}_g)\|^2 \right] , \quad (14)$$

where $\boldsymbol{\epsilon}_{\theta^Q}$ is implemented as a 3-layer MLP with Mish activations; details of implementation can be found in the Appendix C.

To evaluate the policy, the goal-conditioned action-value function is learned with a one-step RL as also proposed by (Gulcehre et al., 2021; Brandfonbrener et al., 2021; Schaul et al., 2015) for offline RL as a stable way of learning Q-functions, minimizing the TD error $\mathcal{L}_{\text{TD}}(\phi)$:

$$\mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D} \left[\left(r^{\mathbf{s}_g} + \gamma Q_{\phi'}(\mathbf{s}', \mathbf{a}', \mathbf{s}_g) - Q_\phi(\mathbf{s}, \mathbf{a}, \mathbf{s}_g) \right)^2 \right], \quad (15)$$

where $\mathbf{a}' \sim \pi_\theta(\mathbf{a} \mid \mathbf{s})$, and the goal-conditioned reward $r(\mathbf{s}, \mathbf{a}, \mathbf{s}_g)$ is defined as the combination of external task reward and intrinsic goal-reaching reward: $r^{\mathbf{s}_g} = r^{\text{ext}}(\mathbf{s}, \mathbf{a}) + r^{\text{intr}}(\mathbf{s}, \mathbf{a}, \mathbf{s}_g)$. In Section 4.4, we conduct an ablation study to examine the impact of intrinsic rewards on our model's performance. Further details regarding the intrinsic reward are available in the Appendix C.

To improve the policy, we minimize the following loss function with hyperparameter α controlling the trade-off between policy extraction and policy improvement as done by (Kumar et al., 2020):

$$\mathcal{L}_q(\theta) = -\alpha \mathbb{E}_{\mathbf{s}\sim D, \mathbf{a}\sim\pi_\theta} \left[Q_\phi(\mathbf{s}, \mathbf{a}, \mathbf{s}_g) \right] . \quad (16)$$

Thus, the final objective for the policy training is given by $\mathcal{L}_\pi(\theta) = \mathcal{L}_d(\theta) + \mathcal{L}_q(\theta)$. In practice, the training of

the Q-Performer can proceed concurrently with that of the D-Conductor, as outlined in Algorithm 1 and Algorithm 2. Empirically, we found that randomly sampling the sub-goal for each state $\mathbf{s}_t$ from the range $[t, t + \Delta * K]$, where $\Delta \sim Geom(p)$, with $p = 0.2$, makes training stable.

3.3. Orchestrating Family

The advantage of hierarchical structures for long-horizon tasks in offline RL is well known (Mandlekar et al., 2020; Pertsch et al., 2020; Rosete-Beas et al., 2023). Therefore, it is worthwhile to investigate alternative hierarchical configurations. Namely PlanQQ, PlanQD, and PlanDD, to further investigate the potential of hierarchical configurations.

In PlanQQ, diverging from PlanDQ, a Q-learning-based conductor, referred to as a Q-Conductor, is utilized at the high level to function as a sub-goal generator: $\pi_\theta(\mathbf{s}_{t+K}|\mathbf{s}_t)$. This Q-Conductor produces a state that is K steps ahead of the current state $\mathbf{s}_t$ as the sub-goal (Park et al., 2023). The Q value function is optimized using the loss function $\mathcal{L}_{\text{TD}}(\phi) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{s}_{t+K}) \sim D} \left[\left(R^K + \gamma Q_{\phi'}(\mathbf{s}_{t+K}, \mathbf{s}') - Q_\phi(\mathbf{s}_t, \mathbf{s}_{t+K}) \right)^2 \right]$, where $R^K = \sum_{i=0}^{K-1} r_{t+i}$. Similar to PlanDQ's Q-Performer, the high-level policy aims to minimize diffusion and policy improvement loss.

PlanQD introduces a variation from PlanQQ by incorporating a Diffuser at the low level to model the sub-trajectory linking state s_t to state s_{t+K}, including actions, as in equation 7. The first action $\mathbf{a}_0$ is executed during RL evaluation.

In PlanDQ, high-level and low-level planners are instantiated as Diffusers, which has been proposed by Chen et al., 2024. A detailed comparison of the experimental performance of these model variants, including PlanDD, is provided in Section 4.4.

Algorithm 2 PlanDQ: Q-Performer Training

1: **Input:** Initialized policy network π_θ, Q-value network Q_ϕ, and target Q-value network $Q_{\phi'}$, offline dataset $\mathcal{D}$
2: **for** each iteration **do**
3: Sample a mini-batch $\mathcal{B} = \{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})\}$
4: Sample a sub-goal $\mathbf{s}_g$ for each $\mathbf{s}_t$
5: Update Q_ϕ by minimizing Equation 15
6: Update policy by minimizing $\mathcal{L}_\pi(\theta)$
7: **end for**=0

4. Experiment

Our experiment section first presents our main results on the standard D4RL (Fu et al., 2020) benchmarks. We also include tasks with extended horizons, specifically designed to assess the long-horizon reasoning capabilities of the model. Following this, we conduct an analysis using a simplified OpenMaze2D environment, offering insights into the reasons behind the superior performance of the Q-learning-based methods compared to value-guided sequence modeling approaches. We end our experiment section with a thorough analysis of our proposed method.

4.1. Experimental Setup

Benchmarks. We first briefly introduce the tasks we used to evaluate our method. The AntMaze suite, known for its challenging long-horizon navigation tasks, is a task where the goal is to control an 8-DoF Ant to reach a pre-set goal from its initial position. Beyond the standard levels of AntMaze included in the D4RL benchmark, our evaluation extends to AntMaze-Ultra (zhengyao jiang et al., 2023), which introduces a larger maze environment. The Kitchen from D4RL and Calvin (Mees et al., 2022) are two long-horizon manipulation tasks. In these two tasks, the agent is required to complete a set of subtasks. The agent will receive a reward of 1 upon the completion of each subtask. The Gym-MuJoCo suite consists of control tasks designed for short-horizon and dense-reward.

Baselines. We consider different classes of baselines that perform well in each domain of tasks. CQL (Conservative Q-Learning, Kumar et al., 2019) is a Q-value constraint offline-RL method. IQL(Implicit Q-Learning, Kostrikov et al., 2022) is a policy regularization-based method. TT (Trajectory Transformer, Janner et al., 2021) utilizes sequence modeling, which leverages the Transformer (Vaswani et al., 2017) to model the entire trajectories. In contrast to TT, Diffuser (Janner et al., 2022a) and DD (Ajay et al., 2022) are two planning-based methods, employing a diffusion probabilistic model (Ho et al., 2020) to model the trajectory distribution. DQL (Diffusion-QL, Wang et al., 2022), meanwhile, utilizes the same diffusion probabilistic model but focuses on learning the behavior policy. Within the realm of hierarchical learning, we consider HIQL (Hierarchical implicit Q-learning, Park et al., 2023), which extends IQL to hierarchical framework. HDMI (Li et al., 2023), DTAMP (Hong et al., 2023b), and HD (Chen et al., 2024) are hierarchical diffusion-based planners. For the purpose of streamlined comparison, we categorize these methods into two groups: flat learning methods and hierarchical learning methods. The source of baseline performances are detailed in Appendix F.

4.2. Long-horizon Navigation and Manipulation

We first evaluate our method on three tasks (AntMaze, Kitchen, Calvin) designed explicitly for long-horizon reinforcement learning. We assess performance based on the ability to achieve task-specific targets, such as reaching the target position in AntMaze or completing all four

Table 2. **Long-horizon Navigation and Manipulation.** PlanDQ combines the benefits of both diffusion-based planning and value-based policy learning, achieving the best performance across all tasks. PlanDQ results are averaged over 100 random planning seeds for AntMaze tasks, and 15 random planning seeds for Kitchen and Calvin tasks.

Environment		Flat Learning Methods				Hierarchical Learning Methods				
		IQL	DQL	Diffuser	DD	HIQL	HDMI	DTAMP	HD	PlanDQ(ours)
AntMaze	Medium	70.0	90.0±3.0	31.9±5.1	0.0	86.8±4.6	–	88.7±2.5	88.7±8.1	**93.0**±2.6
AntMaze	Large	47.5	64.0±4.8	0.0±0.0	0.0	**88.2**±5.3	–	78.0±3.1	83.6±5.8	86.0±3.5
AntMaze	Ultra	21.6	11.0±3.0	0.0±0.0	0.0	52.9±17.4	–	–	53.3±12.9	**70.0**±4.5
AntMaze Average		46.4	55.0	10.6	0.0	76.0	–	–	75.2	**83.0**
Kitchen	Mixed	51.0	62.6	50.0±8.8	65.0	67.7±6.8	69.2±1.8	**74.4**±1.4	71.7±2.7	71.7±2.7
Kitchen	Partial	46.3	60.5	56.2±5.4	57.0	65.0±9.2	–	63.4±8.8	73.3±1.4	**75.0**±7.1
Kitchen Average		48.7	61.6	53.1	61.0	66.4	–	68.9	72.5	**73.4**
Calvin		7.8	N/A	37.5±9.5	40.0±16.6	43.8±39.5	–	–	31.7±8.2	**45.0**±19.8

Table 3. **Offline Reinforcement Learning.** PlanDQ achieves the best overall performance among hierarchical learning methods. PlanDQ results are averaged over 5 random planning seeds. Following Kostrikov et al. (2022), we emphasize in bold scores within 5% of maximum.

Environment Gym Tasks		Flat Learning Methods					Hierarchical Learning Methods			
		CQL	IQL	DD	TT	DQL	DTAMP	HDMI	HD	PlanDQ(ours)
Med-Expert	HalfCheetah	91.6	86.7	90.6±1.3	**95.0**	**96.8**±0.3	88.2	92.1±1.4	92.5±0.3	**95.4**± 0.3
Med-Expert	Hopper	105.4	91.5	**111.8**±1.8	**110.0**	**111.1**±1.3	109.4	**113.5**±0.9	**115.3**±1.1	**111.6**±0.2
Med-Expert	Walker2d	**108.8**	**109.6**	**108.8**±1.7	101.9	**110.1**±0.3	108.2	**107.9**±1.2	**107.1** ± 0.1	**110.2**±0.1
Med-Expert Average		101.9	96.0	103.7	102.3	**106.0**	101.9	104.5	105.0	**105.7**
Medium	HalfCheetah	44.0	47.4	49.1±1.0	46.9	**51.1**±0.5	47.3	48.0±0.9	46.7±0.2	**50.2**±0.1
Medium	Hopper	58.5	66.3	79.3±3.6	61.1	90.5±4.6	80.7	76.4±2.6	**99.3**±0.3	**96.9**±1.3
Medium	Walker2d	72.5	78.3	82.5±1.4	79.0	**87.0**±0.9	82.7	79.9±1.8	84.0±0.6	**86.5**±0.2
Med-Expert Average		58.3	64.0	70.3	62.3	76.2	70.3	68.1	76.7	**77.9**
Med-Replay	HalfCheetah	45.5	44.2	39.3±4.1	41.9	**47.8**±0.3	42.6	44.9±2.0	38.1±0.7	**47.6**±0.1
Med-Replay	Hopper	95.0	94.7	**100.0**±0..7	91.5	**101.3**±0.6	100.0	**99.6**±1.5	94.7±0.7	**101.4**±0.6
Med-Replay	Walker2d	77.2	73.9	75.0±4.3	82.6	**95.5**±1.5	79.5	80.7±2.1	84.1±2.2	**94.0**±0.2
Med-Replay Average		72.6	70.9	71.4	72.0	**81.5**	74.0	75.1	72.3	**81.0**

subtasks in Kitchen and Calvin. As indicated in Table 2, PlanDQ outperforms other methods regarding average task performance, showing a notable 8% improvement over the most robust baseline, HIQL, in the AntMaze task. Additionally, hierarchical learning methods generally outperform flat learning methods in the long-horizon domains. This superiority is attributed to enhanced value estimation, as exemplified in HIQL, or to more efficient long-horizon sequence modeling, as evidenced in HDMI, DTAMP, HD, and our proposed method. Notably, PlanDQ's comparison with DTAMP and HD underscores the effectiveness of utilizing a Q-learning-based diffusion policy at the low level.

4.3. Short-horizon Controlling

We further demonstrate that utilizing a Q-performer at the low level shows enhanced performance over other hierarchical learning methods in short-horizon tasks with dense rewards. As depicted in Table 3, in contrast to the long-horizon tasks discussed previously, nearly all hierarchical learning methods, except PlanDQ, underperform relative to the strongest flat learning method. We exclude HIQL from this comparison as it is specifically designed for goal-conditioned tasks and unsuitable for scenarios without a defined goal. PlanDQ is the only hierarchical learning method that matches the performance of the leading flat learning baselines. Notably, against HD, PlanDQ exhibits a significant improvement of 12% on the Medium-Replay dataset, which consists of a noisy buffer accumulated from a policy trained to the medium agent's level of performance. An overall comparison can be found in Figure 5, where the performance of the baselines is normalized relative to that of PlanDQ. The following section delves deeper into this improvement, and we hope this can offer experimental insights into the underlying reasons for this advancement.

4.4. Analysis and Ablation Study

Diffuser model's value prediction converges to a local optimum in a toy task. To elucidate the reasons for PlanDQ's

improvement over HD on a noisy dataset, we conducted a simplified experiment in the OpenMaze environment, designed to replicate the short-horizon and dense reward setting at a lower level. This environment, showcased in Figure 2, involves guiding a point-mass agent to a fixed goal located at the center of a small open maze. In contrast to a sparse reward setting, the agent receives a reward at each timestep, calculated as the negative exponentiated distance from the goal. To simulate noisiness, we generated a suboptimal dataset using a controller, randomly selecting goal locations while deliberately avoiding the true central goal.

In this setup, we compared Diffusion-QL (DQL) and the Diffuser, the underlying low-level policies in PlanDQ and HD, respectively. The results, presented in Table 4, demonstrate that DQL outperforms Diffuser by a margin of 20%. To delve deeper into the causes of this performance discrepancy, we plotted the estimated value maps for both methods alongside the value map from an optimal policy, as seen in Figure 2. The visual comparison reveals that the learned value function in DQL correctly predicts higher values for states near the center. In contrast, the value predictor in Diffuser erroneously assigns higher values to states away from the center. As a consequence, when guided by its value function, Diffuser tends to lead the agent in suboptimal directions, resulting in less effective behaviors in the task. A potential explanation for this local optimality in Diffuser's value prediction is its learning approach, which involves regressing to the cumulative rewards of trajectories without the Bellman updates. When the horizon is short, this approach tends to introduce errors in value prediction, adversely affecting performance.

Table 4. RL performance on the toy OpenMaze task.

Environment	DQL	Diffuser
OpenMaze	**90.4**±1.9	74.9±1.2

Q-conductor is not stable at the high-level. We verified the benefits of utilizing a Q-performer at the low-level, and it is intriguing to consider whether a Q-learning-based method could similarly enhance the high-level planner. To explore this, we conducted an ablation analysis on various configurations within our hierarchical framework. Specifically, we examined different combinations of diffusion planning and value-based learning for both the high-level conductor and low-level performer. These combinations include PlanDD, which utilizes diffusion planning at both levels; PlanQD, with Q-learning at the high level and diffusion planning at the low level; and PlanQQ, featuring Q-learning at both levels. The normalized scores for these models on the Gym-MuJoCo tasks are presented in Figure 3. The results indicate that employing Q-learning at the high level leads to a decrease in performance on the dense reward tasks. A plausible explanation for this is that the value functions

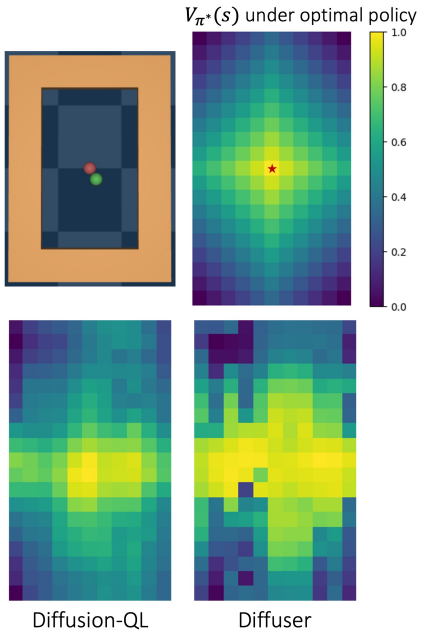


Figure 2. **Value Estimation Comparison.** Compared with Diffusion-QL, Diffuser learns a more noisy value function.

are approximations of expected return with respect to the learning policy, the reward term R^K is collected from the behavior policy, without correction, this makes the Q value function hard to learn with the naive Bellman update. To the best of our knowledge, there is no simple fix within the scope of our problem setting (i.e., offline RL). We will leave this interesting direction for future study.

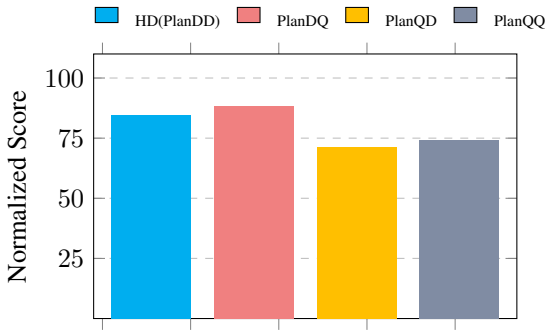


Figure 3. **Model Exploration on the Gym-MuJoCo.** PlanDQ achieves the best averaged performance over all the model variants.

Q-performer outperforms other behavior cloning variants. The low-level performer solves sub-tasks output from the high-level conductor. A goal-conditioned behavior cloning (BC) approach also appears capable of accomplishing this task. To evaluate the necessity and effectiveness of a diffusion-based Q-learning approach, we conduct experiments on the short-horizon Gym-MuJoCo and long-horizon sparse reward AntMaze and Kitchen tasks, compar-

ing against variants with a pure BC performer as well as an agent trained with TD3+BC at the low-level (Fujimoto & Gu, 2021). The results are detailed in Table 5 and Table 7 in the Appendix. The findings indicate that Q-learning performers outperform pure BC performer on these tasks. This superiority may stem from the additional guidance the Q-values provide about the external tasks. Moreover, in comparison with TD3+BC, PlanDQ achieved superior performance due to a more expressive diffusion-based policy.

Table 5. **Ablation Study on Low-Level Performer.** PlanDQ outperforms variants with behavior cloning performers on the Gym-MuJoCo medium replay dataset. Results are averaged over 5 planning seeds.

Tasks	BC	TD3+BC	PlanDQ
HalfCheetah	37.6±0.4	46.2±0.9	**46.7**±0.1
Hopper	23.6±3.1	96.8±1.4	**101.4**±0.6
Walker2d	18.8±2.0	78.9±1.2	**94.0**±0.2
Average	26.7	74.0	**81.0**

External rewards together with intrinsic rewards achieve the best performance. As outlined in earlier sections, our low-level Q-Performer is designed to maximize both cumulative external rewards and intrinsic rewards. In this analysis, we evaluate the effectiveness of this approach by comparing PlanDQ against two variants: Intr-Only, where the low-level Q-Performer is trained exclusively on intrinsic rewards, and Ext-Only, which is trained solely on external rewards. As illustrated in Figure 4, PlanDQ outperforms the other configurations across all dataset qualities. The Intr-Only variant shows weaker performance on the Med-Replay dataset, potentially due to the sub-optimality of the sub-goals generated by the high-level Diffuser. Because it is challenging for Diffuser to learn an optimal value predictor with noisy dataset, as demonstrated in our toy experiments. While the performance of Ext-Only is comparable to that of PlanDQ, PlanDQ notably excels on the Medium dataset.

PlanDQ performs reasonably well with different sub-goal time interval K. Lastly, we assess the robustness of PlanDQ in relation to varying sub-goal time interval K. Intuitively, a larger K may lead to information loss at the high level, resulting in suboptimal sub-goal planning, and could also decrease the efficiency of Q-learning at the low level. Conversely, a smaller K might negate the advantages of a broader receptive field provided by the hierarchical structure (Chen et al., 2024). We conducted experiments on the Gym-MuJoCo and AntMaze tasks with various K values and, due to space constraints, report the Gym-MuJoCo results in Table 6. For the AntMaze results, please refer to the corresponding table in the Appendix. Specifically, we tested $K \in \{2, 4, 8\}$ on the Gym-MuJoCo tasks and

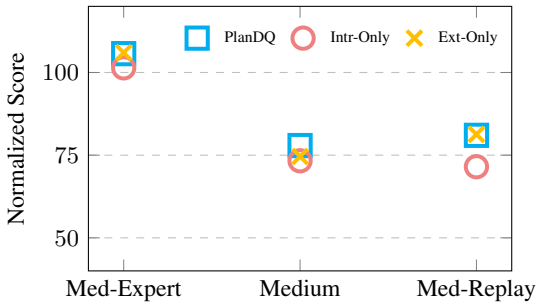


Figure 4. **PlanDQ with different Low-level rewards scheme on the Gym-MuJoCo.** The combination of external rewards with intrinsic rewards achieves the best-averaged performance over different qualities of dataset.

$K \in \{15, 30, 45\}$ on the AntMaze tasks. PlanDQ generally performs well across different K values, albeit with a slight performance drop at $K = 8$.

Table 6. **Ablation Study on Sub-goal Time Interval Step K.** PlanDQ performs reasonably well with different jumpy steps K. Results are averaged over 5 planning seeds.

Environment		K2	K4	K8
Med-Exp	HalfCheetah	**96.1**±0.1	95.4±0.3	94.8±0.6
Med-Exp	Hopper	**112.9**±0.2	111.6±0.2	110.5±0.8
Med-Exp	Walker2d	**110.3**±0.3	**110.2**±0.1	109.9±0.3
Medium	HalfCheetah	49.6±0.8	**50.2**±0.1	49.9±0.5
Medium	Hopper	87.1±1.2	**96.9**±1.3	86.8±1.4
Medium	Walker2d	**86.7**±1.3	**86.5**±0.2	84.9±1.3
Med-Rep	HalfCheetah	**48.1**±0.9	47.6±0.1	47.2±0.7
Med-Rep	Hopper	88.9±0.8	**101.4**±0.6	83.7±2.6
Med-Rep	Walker2d	87.7±1.0	94.0±0.2	**94.8**±0.1
Average		85.2	**88.2**	84.7

5. Related Works

Offline Reinforcement Learning. Offline reinforcement learning (RL) has emerged as a popular approach, allowing agents to learn from pre-existing datasets rather than online interactions with the environment. However, this method presents challenges, such as optimizing policies using limited demonstrations and mitigating overestimation bias caused by the absence of exploration. Value-based offline RL approaches tackle these issues through policy regularization techniques, utilizing Q-learning to achieve Bellman optimality from offline trajectories (Kostrikov et al., 2021; Kumar et al., 2020). These techniques optimize policies to avoid inexperienced actions and evaluate behaviors using learned value functions trained on 'in-sample' data. On the other hand, model-based offline RL approaches adopt a trajectory-planning perspective, utilizing sequence models to generate plausible trajectories (Janner et al., 2021; 2022b; Chen et al., 2021). These model-based approaches have shown promise on tasks with extended horizons. In this paper, we propose a novel hierarchical offline RL approach

that combines the strengths of both methods to leverage their respective benefits.

Diffusion-based Models for Planning. Diffusion-based models have excelled in various domains by effectively modeling complex distributions and generating high-quality samples (Ho et al., 2020; Rombach et al., 2021; Song et al., 2020). In the context of reinforcement learning, diffusion-based models offer valuable applications for planning by incorporating their ability to model world dynamics (Janner et al., 2022b; Ajay et al., 2022) and also notable expressivity(Wang et al., 2023; Chi et al., 2023). Recent works have extended their model to hierarchical architectures (Chen et al., 2024; Wenhao Li, 2023; Hong et al., 2023b), showcasing their effectiveness in scalable environments, including long-horizon scenarios. Our proposed model follows a similar approach, but we present an improved model that outperforms existing models in short-horizon and dense-reward setups while remaining competitive in long-horizon and sparse-reward setups.

Hierarchical Planning. Hierarchical planning problems in RL can be divided into two streams: *parallel* and *sequential* planning. Sequential planning involves predicting future states sequentially, often utilizing recurrent models or temporal generative models (Chung et al., 2017; Kim et al., 2019; Saxena et al., 2021; Hafner et al., 2022) On the other hand, parallel planning aims to compute all timesteps of future horizons simultaneously. This approach commonly employs generative sequence modeling techniques such as Diffuser (Janner et al., 2022b; Ajay et al., 2022). Recent studies have explored the application of the parallel approach in both subgoal plans and low-level trajectories using hierarchical architecture (Li et al., 2023; Chen et al., 2024). These works have demonstrated that their approach achieves superior performance, especially in long-term tasks. However, our study reveals that these methods fall short in short-horizon and dense-reward setups compared to the performance of recent state-of-the-art flat learning models. Based on these observations, our work adopts the parallel approach for high-level subgoal plans and incorporates low-level Q-learning-based models to address the limitations in short-horizon and dense-reward setups.

6. Conclusion and Limitations

In this work, we present PlanDQ, a novel hierarchical planning method for offline reinforcement learning. By leveraging the strengths of diffusion-based sequence modeling and value learning, PlanDQ demonstrates improved performance across a wide range of tasks, excelling in long-horizon and short-horizon settings. Our analysis highlights the limitations of diffusion-based sequence modeling in noisy, short-horizon tasks, where value learning methods perform more effectively. Our findings provide valuable insights into designing efficient offline RL algorithms, paving the way for further advancements in this crucial field.

Limitations and Future Works. Although PlanDQ presents a promising solution for offline RL, our work has several limitations that should be acknowledged.

First, our approach using a diffusion-based high-level planner results in slower policy generation than value-based methods. Future work should focus on improving the efficiency of PlanDQ without sacrificing performance. Second, the quality of the high-level plan in PlanDQ heavily relies on the quality of the data, which could potentially impact the performance. Further investigation into noise-robust high-level planners is necessary to mitigate this limitation. Lastly, our setup utilizes the fixed sub-goal interval. Considering adaptive sub-goal intervals could enhance our method to accommodate the varying characteristics of different tasks.

Impact Statement

The research presented in this paper has the potential to make a significant impact on the fields of reinforcement learning and hierarchical planning. From a theoretical perspective, our work offers a deeper understanding of diffusion-based hierarchical planning within the context of offline reinforcement learning, providing new insights by introducing a novel direction that combines diffusion-based sequence modeling with value function learning methods. On a practical level, the PlanDQ method improves the performance of agents in handling complex tasks, potentially contributing to the development of more proficient autonomous systems in various domains, such as healthcare, autonomous driving, and robotics.

However, with these advancements come potential challenges and responsibilities. As these models become more complex and capable, it is crucial to address the ethical considerations that arise. These include potential biases in decision-making processes, concerns related to data usage and privacy, and the potential displacement of jobs due to increased automation. Ensuring that the ongoing development of offline reinforcement learning is conducted in a responsible and ethical manner is essential as we explore and realize its full potential.

Moreover, it is essential to ensure that the benefits of our work are accessible to all, with a focus on addressing rather than exacerbating existing inequalities and disadvantages faced by marginalized groups. Collaboration among researchers, policymakers, and industry stakeholders is vital for aligning the development and deployment of these models with societal values and fostering a more equitable and inclusive world.

Acknowledgement

This work is supported by Brain Pool Plus (BP+) Program (No. 2021H1D3A2A03103645) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T., and Agrawal, P. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Bhargava, P., Chitnis, R., Geramifard, A., Sodhani, S., and Zhang, A. Sequence modeling is a robust contender for offline reinforcement learning, 2023.

Brandfonbrener, D., Whitney, W., Ranganath, R., and Bruna, J. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.

Chen, C., Deng, F., Kawaguchi, K., Gulcehre, C., and Ahn, S. Simple hierarchical planning with diffusion, 2024.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Chung, J., Ahn, S., and Bengio, Y. Hierarchical multiscale recurrent neural networks. *International Conference on Learning Representations*, 2017.

De Lima, L. M. and Krohling, R. A. Discovering an aid policy to minimize student evasion using offline reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

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

Gulcehre, C., Wang, Z., Novikov, A., Paine, T., Gómez, S., Zolna, K., Agarwal, R., Merel, J. S., Mankowitz, D. J., Paduraru, C., et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

Gulcehre, C., Colmenarejo, S. G., Wang, Z., Sygnowski, J., Paine, T., Zolna, K., Chen, Y., Hoffman, M., Pascanu, R., and de Freitas, N. Regularized behavior value estimation. *arXiv preprint arXiv:2103.09575*, 2021.

Hafner, D., Lee, K.-H., Fischer, I., and Abbeel, P. Deep hierarchical planning from pixels. *arXiv preprint arXiv:2206.04114*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Hong, J., Dragan, A., and Levine, S. Offline rl with observation histories: Analyzing and improving sample complexity, 2023a.

Hong, M., Kang, M., and Oh, S. Diffused task-agnostic milestone planner. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=9Tx2znbyTm.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=wgeK563QgSw.

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

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022b.

Kim, T., Ahn, S., and Bengio, Y. Variational temporal abstraction. *ICML Workshop on Generative Modeling and Model-Based Reasoning for Robotics and AI*, 2019.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*,

2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Kumar, A., Hong, J., Singh, A., and Levine, S. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AP1MKT37rJ.

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

Li, J., Tang, C., Tomizuka, M., and Zhan, W. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4): 10216–10223, 2022.

Li, W., Wang, X., Jin, B., and Zha, H. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, 2023.

Lu, C., Chen, H., Chen, J., Su, H., Li, C., and Zhu, J. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *ICML*, pp. 22825–22855, 2023. URL https://proceedings.mlr.press/v202/lu23d.html.

Ma, Y., Xiao, C., Liang, H., and Hao, J. Rethinking decision transformer via hierarchical reinforcement learning, 2023.

Mandlekar, A., Ramos, F., Boots, B., Savarese, S., Fei-Fei, L., Garg, A., and Fox, D. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420, 2020. doi: 10.1109/ICRA40945.2020.9196935.

Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

Park, S., Ghosh, D., Eysenbach, B., and Levine, S. HIQL: Offline goal-conditioned RL with latent states as actions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=cLQCCtVDuW.

Pertsch, K., Rybkin, O., Ebert, F., Zhou, S., Jayaraman, D., Finn, C., and Levine, S. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems*, 33:17321–17333, 2020.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2021.

Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pp. 234–241. Springer, 2015.

Rosete-Beas, E., Mees, O., Kalweit, G., Boedecker, J., and Burgard, W. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pp. 1838–1849. PMLR, 2023.

Saxena, V., Ba, J., and Hafner, D. Clockwork variational autoencoders. *Advances in Neural Information Processing Systems*, 34:29246–29257, 2021.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

Sinha, S., Mandlekar, A., and Garg, A. S4RL: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8xC5NNej-l_.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Sudhakaran, S. and Risi, S. Skill decision transformer, 2023.

Tang, S., Makar, M., Sjoding, M., Doshi-Velez, F., and Wiens, J. Leveraging factored action spaces for efficient offline reinforcement learning in healthcare. *Advances in Neural Information Processing Systems*, 35:34272–34286, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

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

Wenhao Li, Xiangfeng Wang, B. J. H. Z. Hierarchical diffusion for offline decision making. *Proceedings of the 40th International Conference on machine learning*, 2023.

Yamagata, T., Khalil, A., and Santos-Rodriguez, R. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pp. 38989–39007. PMLR, 2023.

zhengyao jiang, Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cA77NrVEuqn.

A. Benchmark Overview

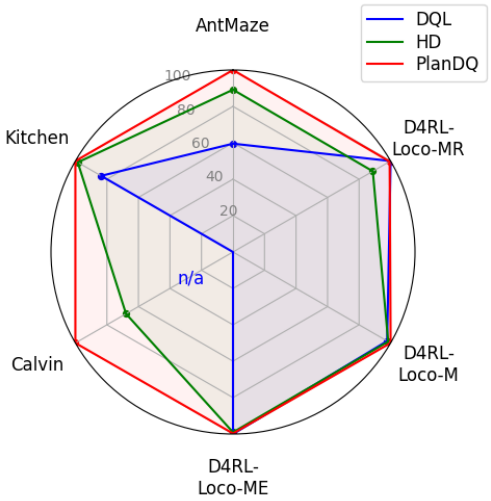
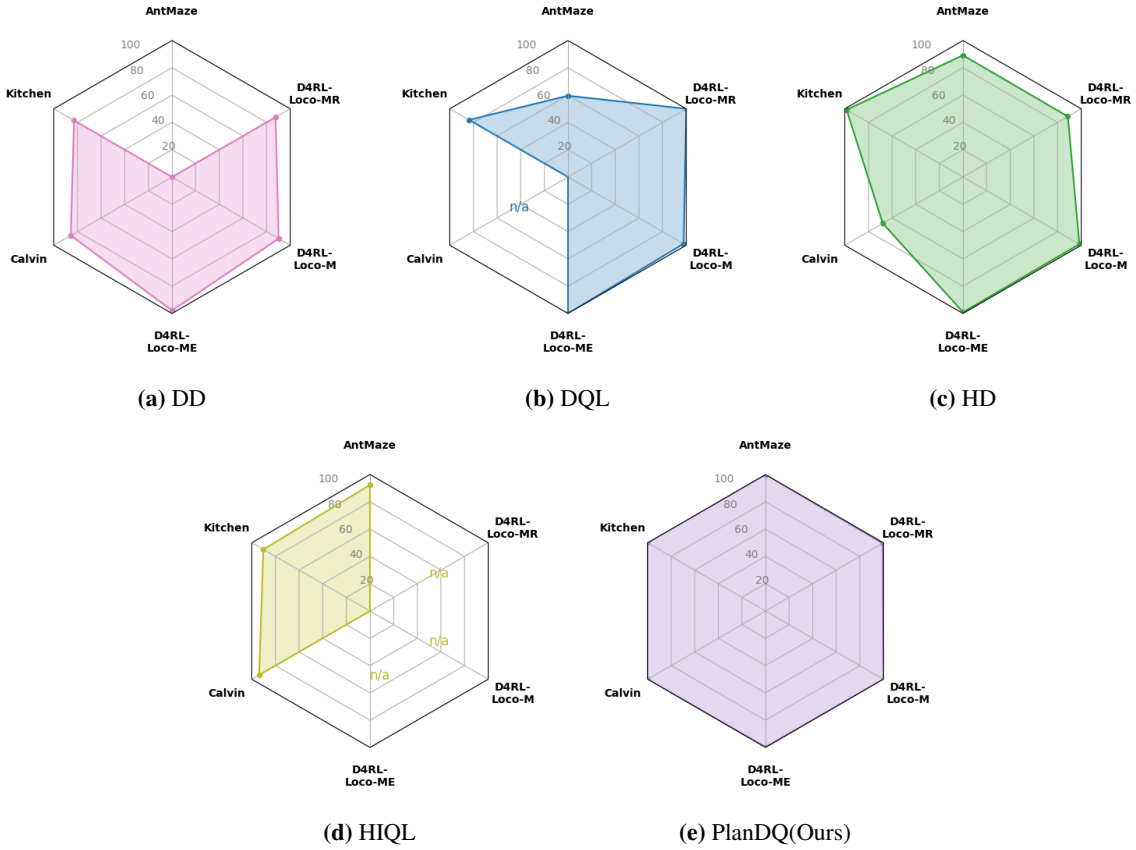


Figure 5. **Benchmark Performance.** The graph presents performance results comparing three main baselines: DQL, HD, and our proposed approach, PlanDQ. The evaluation is conducted across six environments: three long-horizon tasks (AntMaze, Kitchen, and Calvin) and three short-horizon tasks (D4RL-Locomotion-Medium-Replay, Medium, and Medium-Expert). Notably, the scores are normalized to the maximum performance of each benchmark.



(a) DD **(b)** DQL **(c)** HD

(d) HIQL **(e)** PlanDQ(Ours)

Figure 6. Benchmark Overview(Separated Version)

B. An Example-based Analysis for Comparing the Optmality of Diffuser and Q-Learning

Example 1 demonstrates the advantage of the Q-learning over the diffuser for short-horizon problems with dense data. In Example 1, the diffuser degrades as the quality of the data-generating policy decreases, whereas Q-learning learns an optimal policy regardless of the quality of the data-generating policy as long as we have dense data:

Example 1. Consider a MDP, $M = \{S, A, P, R, \gamma, s_0\}$ where $s_0 = c$, $S = \{c\}$ (i.e., $P(c, b, c) = 1$ for $b \in A$), $A = \{b_1, b_2\}$, and $R(c, b_2) > R(c, b_1) > 0$. Then, the optimal policy π^* is obtained by $\pi^*(b_1|c) = 0$ and $\pi^*(b_2|c) = 1$. Let n_1 and n_2 be the numbers of observations of taking action b_1 and b_2 in the training dataset. By chain rule, we can write $p_\theta(\mathbf{x}) = \prod_i p_\theta(\mathbf{s}_i, \mathbf{a}_i|\mathbf{s}_{1:i-1}, \mathbf{a}_{1:i-1}) = \prod_i p_\theta(\mathbf{a}_i|\mathbf{s}_{1:i-1}, \mathbf{a}_{1:i-1}, \mathbf{s}_i)p_\theta(\mathbf{s}_i|\mathbf{s}_{1:i-1}, \mathbf{a}_{1:i-1})$. Thus, in this setting, the global optimizer of the training objective of diffuser $p^*_\theta(\mathbf{x})$ is obtained by $p_\theta(b_i|\mathbf{s}_{1:i-1}, \mathbf{a}_{1:i-1}, \mathbf{s}_i) = \frac{n_i}{n_1+n_2}$ for $i \in \{1, 2\}$ (and $p_\theta(c|\mathbf{s}_{1:i-1}, \mathbf{a}_{1:i-1}) = 1$). By incorporating an arbitrary guidance function, the policy-induced by the optimal diffuser $\hat{\pi}$ is $\hat{\pi}(b_i|c) = \frac{n_i}{n_1+n_2} g(c, b_i)$ for $i \in \{1, 2\}$, where $g(c, b_i)$ is a factor that corresponds to a chosen guidance function. Thus, $\hat{\pi}(b_1|c) > \hat{\pi}(b_2|c)$ if $\frac{n_1}{n_2} > \frac{g(c,b_2)}{g(c,b_1)}$. This shows that for any g satisfying $\frac{g(c,b_2)}{g(c,b_1)} < \infty$, the policy-induced by the optimal diffuser is arbitrarily far away from the optimal policy if $\frac{n_1}{n_2}$ is sufficiently large. Here, the ratio $\frac{n_1}{n_2}$ represents the degree of the suboptimality of the data-generating policy. The condition of $\frac{g(c,b_2)}{g(c,b_1)} < \infty$ is satisfied by common guidance functions (e.g., the value function, the Q-function, the exponential of rewards, etc) since $R(c, b_1) > 0$. On the other hand, the policy induced by the Q-learning is the optimal policy π^*, as long as $n_1, n_2 \geq 1$, regardless of the quality $\frac{n_1}{n_2}$ of the data generating policy. □

More generally, Q-learning is designed to learn a policy better than a data-generating policy (Levine et al., 2020; Fujimoto & Gu, 2021). In contrast, the quality of the diffuser is proportional to the quality of the generating policy since the one factor of the diffuser is designed to maximize the likelihood of the given data. This shows that Q-learner is better, for example, if we have dense data generated by a poor policy.

When we expand all the value functions, the Q-function contains sums over all paths (weighted with the transition probabilities) starting with the current state for the given horizon length. The number of all such paths increases exponentially as the horizon length increases linearly. Thus, the data tends to be denser for shorter horizons. Thus, on the one hand, Q-learner tends to be advantageous for shorter horizons.

On the other hand, the exponential increase in the number of paths means that learning a sufficiently accurate Q-function requires a significantly larger amount of data for a longer horizon. However, the diffuser only fits the distribution of data trajectories instead of the entire tree of such paths. This can result in a good policy without requiring a significantly larger amount of data for a longer horizon when the data-generating policy is not too poor. Thus, diffuser tends to be advantageous for longer horizons.

C. Implementation Details

In this section, we describe the details of implementation and hyperparameters we used during our experiments.

- We build our D-Conductor upon the officially released Diffuser code obtained from https://github.com/jannerm/diffuser.
- We build our Q-Performer with the official code obtained from https://github.com/Zhendong-Wang/Diffusion-Policies-for-Offline-RL
- We set $K = 30$ for the long-horizon planning tasks, while for the Gym-MuJoCo and Kitchen, we use $K = 4$.
- The planning horizon is set to $H \cdot K = 270$ for AntMaze-Medium, $H \cdot K = 450$ for AntMaze-Large, and $H \cdot K = 720$ for AntMaze-Ultra. For short-horizon Gym-MuJoCo, we use $H \cdot K = 32$. For Calvin, the planning horizon is $H \cdot K = 360$. And in Kitchen we use $H \cdot K = 64$
- For the MuJoCo locomotion and Kitchen tasks, we select the guidance scales ω from a set of choices, $\{0.1, 0.01, 0.001, 0.0001\}$, during the planning phase.
- In navigation tasks, the intrinsic reward is calculated as the exponential of the negative Euclidean distance between the state $\mathbf{s}_{t+1}$, reached after executing action $\mathbf{a}_t$, and the sub-goal state $\mathbf{s}_g$: $r^{\text{intr}}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_g) = e^{-\|\mathbf{s}_{t+1} - \mathbf{s}_g\|^2}$. We use the cosineMax proposed in Director (Hafner et al., 2022) as intrinsic reward for other tasks.

D. Additional Experiment Results

D.1. Ablation on the Low-Level Performer Variants

We provide additional results comparing PlanDQ with variants where the low-level performer is trained using behavior cloning methods on the AntMaze-Ultra and Kitchen tasks. As shown in Table 7, PlanDQ outperforms its BC variant due to the expressive diffusion-based policy and Q-learning, which utilizes the external task reward for efficient learning.

Table 7. **Ablation Study on Low-Level Performer.** PlanDQ outperforms variants with behavior cloning performers.

Environment		BC	PlanDQ
AntMaze	Ultra	40.0 ± 15.4	$\mathbf{66.7} \pm 12.2$
Kitchen	Mixed	41.7 ± 5.1	$\mathbf{71.7} \pm 2.7$
Kitchen	Partial	28.3 ± 8.8	$\mathbf{75.0} \pm 7.1$

D.2. Ablation on the Jumpy Steps K

We present the results from PlanDQ with different sub-goal time interval K values on the AntMaze tasks in Table 8. PlanDQ performs well with $K \in 15, 30$. However, performance drops at $K = 45$. This decline may be due to the larger time interval resulting in the loss of key intermediate states, making it challenging for the high-level planner to accurately capture the distribution of sub-goal sequences.

Table 8. **Ablation Study on Sub-goal Time Interval Step K.** PlanDQ performs reasonably well with different interval K. Results are averaged over 15 planning seeds.

Environment		K15	K30	K45
AntMaze	Ultra	60.0 ± 12.6	**66.7** ± 12.2	66.7 ± 12.2
AntMaze	Large	80.0 ± 10.3	**86.7** ± 8.7	60.0 ± 12.6
Average		70.0	**76.7**	63.4

E. Planning with PlanDQ

The details of the classifier-guidance sampling process with PlanDQ are presented in Algorithm 3.

Algorithm 3 Planning with PlanDQ

Input: Current State $\mathbf{s}$, High-level Diffuser $\boldsymbol{\mu}_\theta$, guidance function $\mathcal{J}_\phi$, guidance scale ω, variance σ_m^2, low-level policy π_θ
Initialize plan $\mathbf{x}_M^g \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
while not done **do**
  // Sample the high-level sub-goal plan
  for $m = M-1, \ldots, 1$ **do**
    $\tilde{\boldsymbol{\mu}} \leftarrow \boldsymbol{\mu}_\theta(\mathbf{x}_{m+1}^g) + \omega \sigma_m^2 \nabla_{\mathbf{x}_m^g} \mathcal{J}_\phi(\mathbf{x}_m^g)$
    $\mathbf{x}_{m-1}^g \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \sigma_m^2 \mathbf{I})$
    Fix $\mathbf{g}_0$ in $\mathbf{x}_{m-1}^g$ to current state $\mathbf{s}$
  end for
  // Sample action from low-level policy
  Sample $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{g}_1)$
  Execute action $\mathbf{a}$ in the environment
  Observe state $\mathbf{s}$
end while=0

F. Sources of Baseline Performance

We detailed the sources for our baseline comparisons below:

- **AntMaze Medium and Large**: We sourced the Diffuser and HD results from (Chen et al., 2024). The results for IQL, DD, and DTAMP were taken from (Hong et al., 2023b), and HIQL from (Park et al., 2023). Given that we utilized the version 2 (v2) dataset, and to ensure comparability with the original paper which utilized version 0 (v0), we re-ran DQL with the official code.
- **AntMaze Ultra**: We derived the IQL and HIQL results from (Park et al., 2023). Diffuser, DD, DQL, and HD were assessed using either their official code or the code provided by the authors.
- **Kitchen**: For Kitchen tasks, DQL results were obtained from (Wang et al., 2023), Diffuser and HD from (Chen et al., 2024), IQL and DTAMP from (Hong et al., 2023b), and HIQL from (Park et al., 2023). DD results were sourced from (Ajay et al., 2022).
- **Calvin**: IQL and HIQL results were sourced from (Park et al., 2023). We evaluated Diffuser, DD, and HD using either the official code or the code provided by the authors.
- **Gym Tasks**: Results for CQL, IQL, and DQL were taken from (Wang et al., 2023), HD from (Chen et al., 2024), DTAMP from (Hong et al., 2023a), DD from (Ajay et al., 2022), TT from (Janner et al., 2021), and HDMI from (Li et al., 2022).