Hierarchical Diffusion for Offline Decision Making

Wenhao Li [1 2] **Xiangfeng Wang** [3] **Bo Jin** [4 5] **Hongyuan Zha** [1 2]

Abstract

Offline reinforcement learning typically introduces a hierarchical structure to solve the long-horizon problem so as to address its thorny issue of variance accumulation. Problems of deadly triad, limited data and reward sparsity, however, still remain, rendering the design of effective, hierarchical offline RL algorithms for general-purpose policy learning a formidable challenge. In this paper, we first formulate the problem of offline long-horizon decision-**M**ak**I**ng from the perspective of conditional generative modeling by incorporating *goals* into the control-as-inference graphic models. A **H**ierarchical trajectory-level **D**iffusion probabilistic model is then proposed with classifier-free guidance. HDMI employs a cascade framework that utilizes the reward-conditional goal diffuser for the subgoal discovery and the goal-conditional trajectory diffuser for generating the corresponding action sequence of subgoals. Planning-based subgoal extraction and transformer-based diffusion are employed to deal with the sub-optimal data pollution and long-range subgoal dependencies in the goal diffusion. Numerical experiments verify the advantages of HDMI on long-horizon decision-making compared to SOTA offline RL methods and conditional generative models.

1. Introduction

The fundamentally online learning paradigm of reinforcement learning (RL) hinders its widespread adoption in healthcare, education, finance, and robotics (Charpentier et al., 2021; Sinha et al., 2022; De Lima & Krohling, 2021; Villaflor et al., 2022; Tang et al., 2022) due to the cost and potential harm of online sample collection. Conversely, a large number of samples already logged by operational systems give rise to the problem of offline RL, namely, how to recover high-performing policies without further exploring the environment (Wu et al., 2019; Kumar et al., 2020; Kostrikov et al., 2021; 2022; Ghosh et al., 2022).

Nevertheless, this lack of real-time interactions makes offline RL, in general, notoriously challenging, as value estimators can suffer from potential exponentially increasing variance in terms of the temporal horizon (Li et al., 2015; Ren et al., 2021). Existing work to overcome the "curse of horizon" is either based on less realistic assumptions (Liu et al., 2018; Xie et al., 2019; Ren et al., 2021) or introducing new mechanisms that inherit the high complexity dependency on the horizon (Nachum et al., 2019; Uehara et al., 2020; Yang et al., 2020). The above findings also clearly suggest that it is still an open problem to design an offline RL method that is less affected by the horizon and solves the long-horizon problem effectively.

To deal with long temporal extension, an alternative RL solution is to decompose a long-horizon task into a hierarchy of subproblems, i.e., by hierarchical reinforcement learning (HRL) (Parr & Russell, 1997; Sutton et al., 1999; Kulkarni et al., 2016; Vezhnevets et al., 2017; Nachum et al., 2018), which also sees its utilization in offline decision-making (Ajay et al., 2021; Pertsch et al., 2021b; Villecroze et al., 2022; Rosete-Beas et al., 2022; Rao et al., 2022; Yang et al., 2023). Unfortunately, the instabilities of these methods due to the deadly triad (Sutton & Barto, 2018; Van Hasselt et al., 2018), limited data access (Fujimoto et al., 2019; Kumar et al., 2020), and reward sparsity (Andrychowicz et al., 2017; Ma et al., 2022) still remain largely unresolved.

The success of RL methods leveraging conditional generative models (Chen et al., 2021; Janner et al., 2021; Furuta et al., 2022; Reed et al., 2022; Janner et al., 2022; Ajay et al., 2022; Carvalho et al., 2022) on standardized benchmarks (Fu et al., 2020; Gulcehre et al., 2020) motivates us to consider the following question: *Can the above challenges be mitigated or even avoided using a conditional generation model that introduces a hierarchical structure?*

[1]School of Data Science, The Chinese University of Hong Kong, Shenzhen, China. [2]Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen, China. [3]School of Computer Science and Technology, East China Normal University, Shanghai, China. [4]School of Software Engineering, Tongji University, Shanghai, China. [5]Shanghai Research Institute for Intelligent Autonomous Systems, Shanghai, China. Correspondence to: Xiangfeng Wang <xfwang@cs.ecnu.edu.cn>, Bo Jin <bjin@tongji.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

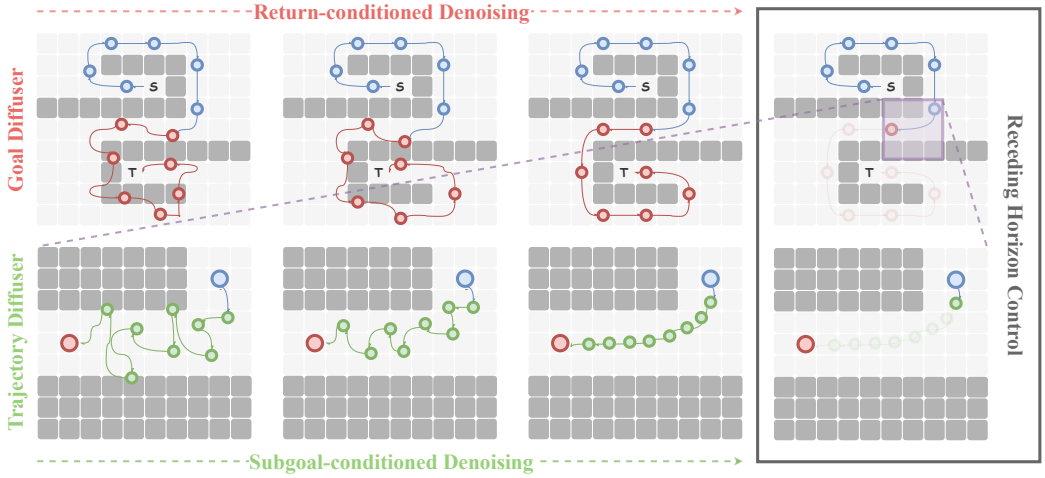


Figure 1: Illustrative example of hierarchical diffusion. We first sample the subgoals based on the return and then samples the actions corresponding to the subgoal. We use receding horizon control to avoid the error accumulation due to stochastics.

This paper provides an affirmative answer algorithmically and empirically. Our core idea follows offline goal-based methods (Ma et al., 2021; Liu & Sun, 2022; Ma et al., 2022; Li et al., 2022) based on constructing a set of subproblems by discovering subgoals and further learning one (unified) policy to reach these subgoals (Pateria et al., 2021; Uehara et al., 2020). We first formulate the offline long-horizon decision-making as the goal-based, *conditional generative modeling* by introducing the *goal* into the control-as-inference framework (Levine, 2018) and estimating the *goal-action* joint conditional distribution with the hierarchical diffusion. The generative learning operates in a different spirit from temporal difference learning; thus, there are fewer influences by the deadly triad and limited dataset.

Concretely, the proposed **H**ierarchical **D**iffusion for Offline Decision **M**ak**I**ng framework (Figure 1), **HDMI**, employs a cascade framework similar to independent subtask discovery (Pateria et al., 2021, §3.3) in HRL, where the *goal diffuser* learns a reward-conditional model for subgoal discovery, and the *trajectory diffuser* learns a goal-conditional model for action generation. The goal diffuser learns from high-quality subgoals, and we adopt a planning-based method (Eysenbach et al., 2019) to avoid the pollution caused by sub-optimal data. Furthermore, previous work has shown that the correlations between elements (e.g., image pixels (Song et al., 2021) or trajectory states (Janner et al., 2022)) are critical to the success of diffusion models. Therefore, the weakness of dependencies between local subgoals and long-range dependencies between global subgoals motivate us to utilize the transformer-based diffusion model (Vaswani et al., 2017; Peebles & Xie, 2022) instead of commonly used U-Net-like (Ronneberger et al., 2015) structure. Moreover, goal-conditional policy learning is translated into an inpainting problem (Sohl-Dickstein et al., 2015; Janner et al., 2022) by replacing the sampled states with conditioning subgoals for cascade trajectory diffuser.

Using a diffusion probability model has several significant advantages for solving long-horizon problems over existing generative models families: The polylogarithmic dependency on the horizon (or dimensionality Lee et al. (2022a)) and GAN-level sample quality without adversarial training enables efficient long-horizon modeling; the flexible model architecture enables transformer-based sparse subgoal dependencies capturing; inverse problem-solving without retraining models enables practical subgoal discovery and action generation. Moreover, the diffusion model blurs the line between modeling and planning, allowing the model to improve the prediction while improving the planning ability during training, thus better solving the credit assignment planning problem due to sparse rewards (Janner et al., 2022).

Our main contributions include: 1) introducing *goals* into the control-as-inference framework and formulating offline long-horizon decision-making as a conditional generative modeling; 2) employing a hierarchical framework, HDMI, in which the goal diffuser learns a reward-conditional diffusion for the subgoal discovering, and the trajectory diffuser learns a goal-conditional diffusion for the action generation; 3) utilizing a planning-based subgoal extractor and transformer-based diffusion to deal with the sub-optimal data pollution and long-range subgoal dependency issues.

2. Problem Formulation

To take advantage of the generative model, we need to transform the offline long-horizon decision-making, i.e., obtaining the most probable subgoal and action distributions that maximize the expected return, into a goal-based, conditional

generative modeling problem by maximizing:

$$\mathbb{E}_{\boldsymbol{\tau}_0 \sim \mathcal{D}}\big[\log p_\theta\big(\boldsymbol{\tau}_0 \mid y(\boldsymbol{\tau}_0)\big)\big], \tag{1}$$

where $\boldsymbol{\tau}_0 = \{g_i, \{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}\}_{i=1}^{N-1}$ represents a trajectory with a hierarchical structure in the dataset. Note that we have $g_i \equiv s_{i*M}, i \in [0, N-1]$, and each trajectory must reach all subgoals exactly. In other words, each trajectory is generated under the subgoal constraints. $\{g_i\}_{i=0}^{N-1}$ represents the subgoal sequence of length N, and $\{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}\}$ represents the trajectory of length M corresponding to subgoal g_i, the horizon is $T := N \times M$. Then the goal of offline long-horizon decision-making is to estimate the conditional data distribution with p_θ so we can generate a target trajectory $\boldsymbol{\tau}_0$ from the information $y(\boldsymbol{\tau}_0)$.

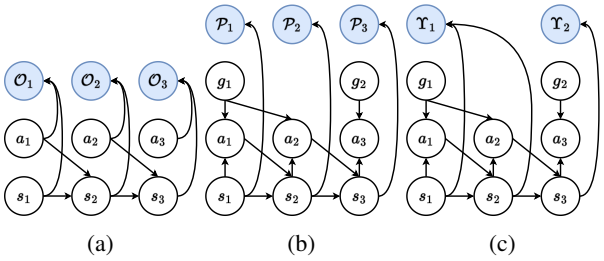


Figure 2: Graphic models for decision-making: (a) The states and actions form the backbone of the graphic model; (b) Augmenting goals to embed a long-horizon problem into this graphic model; (c) By summarizing the goal-augmented optimality variables, the new goal-oriented graphic model more naturally models the probability distribution of subgoals and highlights the hierarchical structure. Similar with (Levine, 2018), we condition the optimality variables as being *true* and then infer the most probable higher-level subgoal and lower-level action sequence or distributions.

The goal of long-horizon offline decision making is to extract knowledge from a fixed offline dataset to obtain a policy that can maximize cumulative rewards[1]. And the goal-based mechanism we follow is to achieve the above goals by continuously achieving sub-goals through constraint policies. However, the current log-likelihood in Equation 1 has not yet been linked to rewards and sub-goals. To this end, we first introduce the control-as-inference framework (Levine, 2018), and its corresponding probabilistic graphical model is shown in Figure 2a (refer to Appendix B.3).

Although the control-as-inference framework can incorporate reward into the goal-based condition generation problem, it does not leave a place for subgoal. To do further analysis and conversion, we introduce subgoal sequences into the basic graphical model of control, as shown in Figure 2b. Adding the subgoal variables g_i results in a new graphical model, which can be conditioned on some new optimality variables $\mathcal{P}_t$ that can represent either the same subgoal or a different subgoal. This new optimality variable is a binary random variable, where $\mathcal{P}_t = 1$ denotes that the state-action pair conditioned on the subgoal g_i is optimal, otherwise $\mathcal{P}_t = 0$. We can then naturally define the information $y(\boldsymbol{\tau}_0)$ as the optimality variables $\mathcal{P}_{0:T-1}$ of the trajectory $\boldsymbol{\tau}_0$ being *true*.

This "goal-augmented" graphical model in Figure 2b has a semantically identical interpretation as the original graphical model in Figure 2a, where the combination of the transition model and the goal conditional serves as a new (and likely more manageable) dynamical system. In other words, the lower-level trajectory shapes the underlying dynamics of the system, ideally making it more easily controllable by a higher-level subgoal sequence.

A specific data distribution for the higher-level subgoals and lower-level trajectory can be learned by conditioning on new optimality variables hierarchically. Before that, we transform the graphical model in Figure 2b into a more compact one to naturally model the probability distribution of subgoals and highlight the hierarchical structure of the long-horizon problem. Specifically, the compact graphic model, as shown in Figure 2c, conditions on the new goal-based optimality variable Υ_i by summarizing $\mathcal{P}$, which is also a binary random variable. $\Upsilon_i = 1$ denotes that the generated trajectory of subgoal g_i is optimal, otherwise $\Upsilon_i = 0$. Then the information $y(\boldsymbol{\tau}_0)$ is correspondingly transformed to the goal-based optimality variables $\Upsilon_{0:N-1}$ of the trajectory $\boldsymbol{\tau}_0$ being *true*, where $g_{0:N-1} \in \boldsymbol{\tau}_0$.

Further, without loss of generality, we introduce the following definition to formalize the posterior distribution of goal-based optimality variables similar with (Levine, 2018):

Definition 2.1 (Posterior distribution of goal-based optimality variable)**.** Given a subgoal g_i and its corresponding trajectory $\{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}$, the posterior distribution of the goal-oriented optimality variable Υ_i can be modeled as an energy-based model as follows:

$$p\left(\Upsilon_i = 1 \mid g_i, \{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}\right) \propto \exp\left(\sum_{j=1}^{M} r\left(s_{(i-1)*M+j}, a_{(i-1)*M+j}\right)\right).$$

At this point, we can link Equation 1 with the optimization objective of offline long-horizon decision-making. Concretely, we can expand the conditional probability in Equation (1) into the following hierarchical form:

[1]Achieving a desired goal can also be modeled as a reward maximization task, except that the agent is rewarded only after reaching the goal state.

Proposition 2.2. *Given a subgoal sequence $\tau_g := g_{0:N-1}$ and the corresponding trajectory $\tau_{sa} := \{s_{0:T-1}, a_{0:T-1}\}$, $T = N * M$, the conditional probability in Equation (1) can be transformed into following form based on the graphic model shown in Figure 2c:*

$$p\left(\boldsymbol{\tau}_0 \mid y(\boldsymbol{\tau}_0)\right) \propto p(\tau_g) y(\tau_g) \prod_{i=1}^{N} p(\tau_{sa}^i) y(\tau_{sa}^i), \tag{2}$$

where $\tau_{sa}^i := \{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}$ corresponding to the subgoal g_i, $y(\tau_g) := \exp\left(\sum_{t=0}^{T-1} r(s_t, a_t)\right)$ and $y(\tau_{sa}^i)$ is a Dirac delta for the subgoal constraints.

With Proposition 2.2, we transform offline long-horizon decision-making into the goal-based, conditional generative modeling with a unique hierarchical form. Given a decision-making task, to sample an optimal trajectory, we first conditionally sample the optimal subgoal sequence τ_g based on the information $y(\tau_g)$; and then conditionally samples the optimal trajectory τ_{sa}^i corresponding to the subgoal g_i based on the information $y(\tau_{sa}^i)$. We will further introduce how to construct the training dataset, parameterize and train the model based on a fixed offline dataset, and finally sample our interested subgoal and action sequences from it.

It is noteworthy that the formalism of goal-conditioned control of inference has garnered prior attention (Rudner et al., 2021). However, we depart from the previous literature regarding the problem's objective, modeling approach, and solution. Rudner's formulation represents the problem of goal-conditioned, online RL as a novel policy inference task aimed at achieving a desired outcome, while our approach casts the long-horizon offline decision-making issue as a policy inference matter that focuses on maximizing rewards. The introduction of goals serves only to establish a hierarchical structure for more efficient problem-solving.

These differences also yield distinct strategies for addressing the problem. Rudner's approach necessitates the acquisition of a goal-related reward function to implement an online RL framework, given that rewards are not included in the modeled problem. In contrast, our proposed method entails transforming policy inference into a reward- and goal-conditioning generation problem and resolving it through maximum likelihood estimation based on an offline dataset. Please refer to Appendix G for more analysis.

3. The Proposed Method

This section discusses how we may use a hierarchical diffusion model for above conditional sampling. First, we discuss how to extract high-quality subgoal from datasets with variable sample quality in Section 3.1. Next, we discuss the modeling choices for hierarchical diffusion in Section 3.2. We then discuss how we may train our models in Section 3.3 and utilize a hierarchical classifier-free structure to capture the best aspects of subgoals and actions in Section 3.4.

3.1. Planning-based Subgoal Extraction

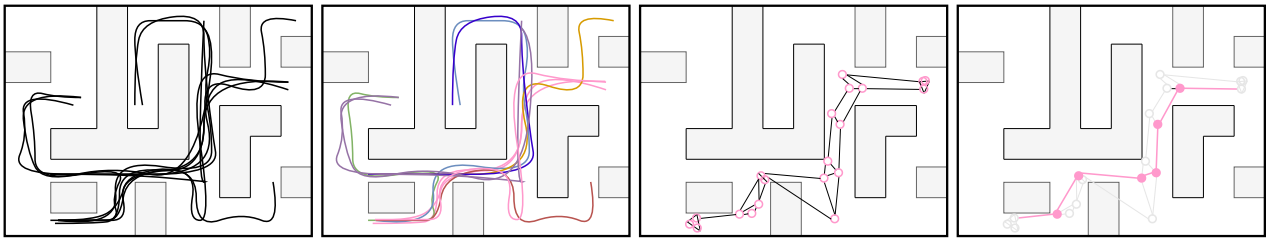
Proposition 2.2 shows that the lower-level action generation depends only on the subgoal sequences generated in the upper level, independent of the reward function. The reward function only affects the generation of subgoal sequences. Therefore, the selection of the subgoal prior distribution $p(\tau_g)$ in (2) is critical to generate high-quality action sequences. Unfortunately, in the offline dataset, no subgoals correspond to each trajectory in advance. This requires us to preprocess the dataset and extract high-quality subgoal sequences. The critical challenge is that suboptimal trajectories pollute the dataset. For example, in a goal-reaching task, the two trajectories to the goal may differ significantly in length. Thus extracting the corresponding subgoals from each trajectory *independently* will not guarantee optimality.

To this end, we borrow a planning-based online RL method, SoRB (Eysenbach et al., 2019), which can automatically find subgoals by learning graphic abstractions of the environment (Figure 3). This graph is constructed via an extra RL task, where a goal-conditioned value function provides edge weights, and nodes are taken to be observations. Using graph search to find the shortest path, we can automatically generate subgoals, even in high-dimensional environments.

Specifically, we define an additional goal-conditioned reward function that returns -1 at each step, i.e., $r(s_t, a_t, s_{t+1}) = -1$. This leads to a close connection between the goal-conditioned value function and shortest paths. That is, the value of state s_1 with respect to any other state s_2 is simply the negative shortest path distance (Eysenbach et al., 2019): $V(s_1, s_2) = d_{sp}(s_1, s_2)$, where $d_{sp}(\cdot, \cdot)$ is the short path distance between two states. We choose the off-policy RL algorithm with goal relabelling to learn the goal-conditioned value function and introduce distributional RL (Morimura et al., 2010; 2012; Bellemare et al., 2017) to improve the accuracy of distance estimation. The IQN (Dabney et al., 2018) for discrete actions and the D4PG (Barth-Maron et al., 2018) for continuous actions are employed, while both operate over transitions sampled from the offline dataset $\mathcal{D}$. Further, we employ the learned goal-conditioned value function to construct a weighted, directed graph: $\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V} = \mathcal{D}$, $\mathcal{E} = \mathcal{D} \times \mathcal{D} = \{e_{s_1 \to s_2} \mid s_1, s_2 \in \mathcal{D}\}$, and

$$\mathcal{W}\left(e_{s_1 \to s_2}\right) = \begin{cases} V\left(s_1, s_2\right) & \text{if } V\left(s_1, s_2\right) < \Delta_g; \\ \infty & \text{otherwise,} \end{cases} \tag{3}$$

where Δ_g denotes a hyperparameter controlling the length of the subgoal sequence. However, due to the online charac-



(a) Initialization. (b) Clustering. (c) Graph Construction. (d) Planning.

Figure 3: Planning-based subgoal extraction. For the original dataset $\mathcal{D}$, we first cluster all trajectories using the mini-batch k-means++ algorithm to aggregate trajectories with the same initial and the terminal state into a sub-dataset $\mathcal{D}_c$; next, we transform $\mathcal{D}_c$ based on the distributional off-policy RL to a weighted directed graph $\mathcal{G}_c$, where the nodes represent the states and the weights represent the predicted shortest distance; finally, we use the proposed offline version of (Eysenbach et al., 2019) to search for the shortest path from the initial to the terminal state on $\mathcal{G}_c$ and obtain the subgoal sequence.

teristics of SoRB, directly migrating it to an offline setting will make the problem intractable. There are two reasons: 1) in SoRB, subgoals are generated online. SoRB will research for the next subgoal every time the agent interacts with the environment and transits to the next state; 2) SoRB needs to use the Floyd-Warshall algorithm (Floyd, 1962; Roy, 1959; Warshall, 1962) to calculate the shortest path between every two states in the dataset in advance with best-case performance $O\left(|\mathcal{D}|^3\right)$. This makes it challenging to scale to offline datasets with millions of states.

To address the two challenges, we try to reduce the dataset size by clustering. Considering the manifold of samples in the offline datasets used in this paper, we use the mini-batch version (Sculley, 2010) of k-means++ (Arthur & Vassilvitskii, 2007) for clustering. Trajectories whose initial and final state belong to the same centroid will be collected as $\mathcal{D}_c$ and used to construct the sub-graph G_c.

Algorithm 1 Next Subgoal Searching on the Sub-graph.

Input: the current goal s_g, the terminate state s_T, dataset $\mathcal{D}_c$, the learned goal-conditioned value function V.
$M_{\pi_b} \leftarrow -V(\mathcal{D}_c, \mathcal{D}_c)$; ▷ cached
$M_{\mathcal{D}_c \to \mathcal{D}_c} \leftarrow \text{FloydWarshall}\left(M_{\pi_b}\right)$; ▷ cached
$M_{s_g \to \mathcal{D}_c} \leftarrow -V(s_g, \mathcal{D}_c)$; $M_{\mathcal{D}_c \to s_T} \leftarrow -V(\mathcal{D}_c, s_T)$;
$M_{s_g \to s_T} \leftarrow M_{s_g \to \mathcal{D}_c} + M_{\mathcal{D}_c \to \mathcal{D}_c} + \left(M_{\mathcal{D}_c \to s_T}\right)^{\mathrm{T}}$;
$u, v \leftarrow \arg\min_{u,v \in \mathcal{D}_c} M_{s_g \to s_T}$;
Output: the next subgoal $s_{g'} := u$.

An offline version of (Eysenbach et al., 2019) is further utilized to obtain a *shared* subgoal sequence for each centroid (Algorithm 1). $M_{\pi_b}, M_{\mathcal{D}_c, M_{s_g \to s_T} \to \mathcal{D}_c} \in \mathbb{R}^{|\mathcal{D}_c| \times |\mathcal{D}_c|}$ are matrices, $M_{s_g \to \mathcal{D}_c}, M_{s_g \to s_T} \in \mathbb{R}^{|\mathcal{D}_c|}$ are vectors. Ignoring the pre-cached $M_{\pi_b}, M_{\mathcal{D}_c}$, Algorithm 1 only needs to call the learned value function $O\left(|\mathcal{D}_c|\right)$ times. Then, for each trajectory in $\mathcal{D}_c$, we find the state closest to each subgoal and replace it with that subgoal. This makes each subgoal sequence correspond to a set of diverse trajectories, which will facilitate the model training and sampling diversity.

3.2. Transformer-based Hierarchical Diffusion

After preprocessing the dataset, we get the dataset $\mathcal{D} := \{\boldsymbol{\tau}_0\}$ contains the sample with hierarchical structure $\boldsymbol{\tau}_0 = \{g_i, \{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}\}_{i=1}^{N-1}$ and subgoal constraints $g_i \equiv s_{i*M}, i \in [0, N-1]$. We can then parameterize Equation 2 by using a hierarchical diffusion probability model. Nevertheless, before that, we need to choose a suitable skeleton of the diffusion model.

Existing work has shown that the capture of correlations between elements (e.g., image pixels (Song et al., 2021) or trajectory states (Janner et al., 2022)) is critical to the success of diffusion models. Different from related works that use the diffusion model to denoise the entire state trajectory or state-action trajectory, we denoise the *sparser* subgoal trajectory in the goal diffusion. The long-range dependence between subgoals makes the U-Net-like (Ronneberger et al., 2015; Janner et al., 2022; Ajay et al., 2022) structure based on local convolution no longer the optimal choice. This motivates us to use the transformer (Vaswani et al., 2017; Peebles & Xie, 2022) as the skeleton of the diffusion model.

Specifically, we use a transformer-based neural network as shown in Figure 4 to parameterize $\epsilon_\theta\left(\boldsymbol{\tau}_k, y(\boldsymbol{\tau}), k\right)$ (see §B.2) for both goal diffusion and trajectory diffusion[2]. In the denoising process of step $k+1$, we first linearly encode each subgoal or state in the noised trajectory obtained in the previous step k, then concatenate the standard ViT (Dosovitskiy et al., 2020) frequency-based sine-cosine positional embeddings. In addition to noised trajectory inputs, our hierarchical diffusion model needs to process conditioning information, i.e., timesteps k and information $y(\boldsymbol{\tau}_k)$. Sim-

[2](Peebles & Xie, 2022) has shown that the transformer is comparable to U-Net. Therefore, we uniformly use the transformer-based diffusion to reduce the complexity.

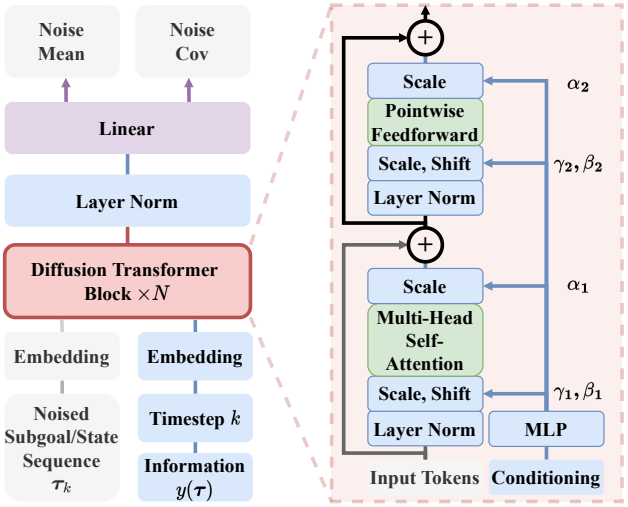


Figure 4: Transformer-based hierarchical diffusion model. Each elements in the noised trajectory is first linearly embedded and then concatenated with frequency-based sine-cosine positional embedding. Information $y(\boldsymbol{\tau_k})$ and timestep k are also linearly embedded first and then regarded as two additional tokens that are appended to the noised trajectory embedding. The augmented input tokens are then processed by a sequence of diffusion transformer blocks (Peebles & Xie, 2022). To fully utilize conditional information, the embedding of information $y(\boldsymbol{\tau_k})$ and timestep k will undergo a linear embedding to generate adaptive dimension-wise scaling and shift parameters γ, β, α, with adaptive layer norm and linearly decoding into an noise prediction and an diagonal covariance prediction.

ilar to cls tokens in ViTs, k and $y(\boldsymbol{\tau_k})$ are also linearly embedded first and then regarded as two additional tokens are appended to the trajectory embedding. For the goal diffuser, y represents the normalized return ($y \in [0, 1]$) of the subgoal sequence. y represents the subgoal constraints in the trajectory diffuser, and we will discuss it soon.

The input tokens are then processed by multiple diffusion transformer (DiT) blocks (Peebles & Xie, 2022). The DiT block replaces standard layer norm layers with adaptive layer norm. Rather than directly learn dimension-wise scale and shift parameters γ and β, DiT regresses them from the sum of the embedding vectors of k and $y(\boldsymbol{\tau_k})$ to make full use of conditional information. The DiT block also regresses dimension-wise scaling parameters α applied immediately before any residual connections within the DiT block.

3.3. Model Training with Classifier-free Guidance

Recall that we want to approximate $p\left(\boldsymbol{\tau}_0 \mid y(\boldsymbol{\tau}_0)\right)$ by using a hierarchical diffusion model from $\mathcal{D} := \{\boldsymbol{\tau}^i\}$. After parameterizing $\epsilon_\theta\left(\boldsymbol{\tau}_k, y(\boldsymbol{\tau}), k\right)$, we can then train the model and sample from it. In the goal diffuser, for each sampled trajectory $\boldsymbol{\tau}_{g,0}$, we first sample noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a timestep $k \sim \mathcal{U}\{1, \ldots, K\}$, and construct a noised subgoal sequence $\boldsymbol{\tau}_{g,k} := \sqrt{\alpha_k}\boldsymbol{\tau}_{g,0} + (1-\alpha_k)\epsilon$. The unconditional noise is represented as the conditional noise $\epsilon_{\theta_g}\left(\boldsymbol{\tau}_{g,k}, \varnothing, k\right)$ where a dummy value $\varnothing$ takes the place of $y(\boldsymbol{\tau}_{g,k})$. We then predict the noise as $\hat{\epsilon}_{\theta_g} := \epsilon_{\theta_g}\left(\boldsymbol{\tau}_{g,k}, (1-\beta)y(\boldsymbol{\tau}_{g,k}) + \beta\varnothing, k\right)$, with probability p_u we ignore the conditioning information. The corresponding training loss is:

$$L_G(k, \boldsymbol{\tau}_{g,0}, p_u) = \mathbb{E}_{k, \boldsymbol{\tau}_{g,0} \in \mathcal{D}, \beta \sim \mathrm{Bern}(p_u)}\left[\left\|\epsilon - \hat{\epsilon}_{\theta_g}\right\|^2\right],$$

where the condition $y(\boldsymbol{\tau}_{g,k})$ is the maximum value of the normalized return in the trajectories corresponding to $\boldsymbol{\tau}_{g,0}$.

The trajectory diffuser is trained similarly with a few trade-offs. Firstly, trajectory diffuser can model $p(\boldsymbol{\tau}^i_{sa,0}|y(\boldsymbol{\tau}^i_{sa,0}))$, where $\boldsymbol{\tau}^i_{sa,0}$ represents the sub-trajectory $\{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}$ corresponding to a certain subgoal g_i. We can naturally choose to corrupt and denoise directly to $\boldsymbol{\tau}^i_{sa,0}$ just like (Janner et al., 2022). However, sequences over actions tend to be more high-frequency and less smooth, thus making it harder for the diffusion model to predict (Kingma et al., 2021; Tedrake, 2022).

Therefore, we only use the trajectory diffuser to estimate the posterior distribution of state trajectories $\boldsymbol{\tau}^i_{s,0} := \{s_{(i-1)*M+j}\}_{j=1}^{M}$, and use inverse dynamics model (Agrawal et al., 2015; Pathak et al., 2017) similarly to (Ajay et al., 2022) in order to generate action trajectories based on state trajectories. Training this inverse dynamics model amounts to learning function F defined as:

$$\hat{a}_{(i-1)*M+j} = F\left(s_{(i-1)*M+j}, s_{(i-1)*M+j+1}; \theta_I\right),$$

where $\hat{a}_{(i-1)*M+j}$ is the predicted estimate of action $a_{(i-1)*M+j}$ and the parameters of inverse dynamics model θ_I are trained to optimize $L_I\left(\hat{a}_t, a_t; \theta_I\right)$. In case $a_{(i-1)*M+j}$ is discrete, F's output is a softmax distribution and minimizing L_I amounts to maximum likelihood estimation of θ_I under a multinomial distribution; Otherwise, L_I is instantiated as the mean squared error.

Furthermore, the conditioning information $y(\boldsymbol{\tau}^i_{sa,0})$ is a Dirac delta for the subgoal constraints as mentioned in Proposition 2.2. How constraints are fed into a diffusion model is not trivial. Fortunately, this setting can be reformulated into an *inpainting* problem, in which state and action constraints act analogously to observed pixels in an image (Janner et al., 2022). This can be practically implemented by sampling from the denoising process $\boldsymbol{\tau}^i_{s,k-1} \sim p_{\theta_s}\left(\boldsymbol{\tau}^i_{s,k-1} | \boldsymbol{\tau}^i_{s,k}\right)$ and replacing the last state of the sampled trajectory with conditioning subgoal g_i after each reverse timesteps k. Interestingly, this means that when training the trajectory diffuser, we do not need to input the conditioning information $y(\boldsymbol{\tau}^i_{sa,0})$ but only need to use the above trick when generating trajectories during the

test phase. Similar to the goal diffuser, the training loss is:

$$L_T(k, \boldsymbol{\tau}^i_{s,0}) = \mathbb{E}_{k, \boldsymbol{\tau}^i_{s,0} \in \mathcal{D}} \left[\|\epsilon - \hat{\epsilon}_{\theta_s}\|^2 \right], \tag{4}$$

where $\hat{\epsilon}_{\theta_s} := \epsilon_{\theta_s}(\boldsymbol{\tau}^i_{s,k}, \varnothing, k)$. The training procedure of the goal and trajectory diffuser is shown in Algorithm 2. In the algorithm, $\boldsymbol{\tau}_\cdot$ is the collective name of $\boldsymbol{\tau}_{g,\cdot}$ and $\boldsymbol{\tau}_{s,\cdot}$, and θ is the collective name of θ_g and θ_s.

Types of problems that HDMI can solve. The model training so far is mainly aimed at the reward-maximizing. That is, the conditional information of the goal diffuser trajectory is related to the reward. However, HDMI is equally applicable to solving the goal-reaching problem (i.e., the final state is constrained by the goal), simply by applying the training method of the trajectory diffuser to the goal diffuser, as shown in Algorithm 4.

3.4. Decison-Making with Hierarchical Diffusion

After the model is trained, sampling from HDMI is equivalent to complete the planning in RL. Reward-maximizing decision-making with the hierarchical diffusion is shown in Algorithm 3, and for the goal-reaching task, only a simple modification is required, as shown in Algorithm 5. It is worth noting that the generation of action does not depend on the reward. And in the subgoal extraction, the same subgoal may correspond to multiple action sequences of varying quality. This makes the trajectory diffuser, while capable of sampling diversity, to potentially generate low-quality data. For this reason we borrowed the low-temperature sampling in (Ajay et al., 2022) to avoid this problem.

There are two additional points need to be explained: the classifier-free guidance and the receding horizon control (RHC). For the former we use a discrete-time version of (Ho & Salimans, 2022). And for the latter, due to the aleatoric uncertainty of the environment and the epistemic uncertainty of the model, sampling the whole subgoal and action sequence in the test phase will lead to errors that are exponentially proportional to the planning horizon.

A standard solution in the optimal control, as well as in RL, is to use model predictive control (MPC). However, existing work generally uses fixed horizon optimization. This means we first generate a sequence of length N or M using the goal or trajectory diffuser and only use the first element of the sequence as the sampling result. Then, the model starts from this element and repeats the above operation until the length of the sampled sequence meets the requirement.

However, this method ignores the sequence that has been sampled and only relies on the result of the previous sampling step for subsequent sampling. This prevents the advantages of transformer-based diffusion models in modeling long-range dependencies from being fully exploited. Thus, we adopt a RHC method subordinate to MPC similar to (Ajay et al., 2022). RHC does not perform fixed horizon optimization like MPC. The horizon at each sampling step is one less than the previous step, but it relies on one more step information than the previous step, as shown in Algorithm 3. To utilize the already sampled information, we employ a similar inpainting trick to that used in the trajectory diffuser.

4. Experiments

This section aims to verify the effectiveness of HDMI in long-horizon goal-reaching, reward-maximizing, and realistic tasks. We emphasize in bold scores within 5 percent of the maximum per task (Kostrikov et al., 2022).

4.1. Long-Horizon Goal-Reaching

To visually verify the advantage of HDMI on long-horizon decision-making tasks, we use the Maze2D and AntMaze dataset (Fu et al., 2020). In these tasks, the agent only receives a $+1$ reward when it gets close to the goal location.

Baselines: MPPI (Williams et al., 2016) is a sampling-based MPC algorithm using ground-truth dynamics; CompILE (Kipf et al., 2019) is a hierarchical imitation learning method based on joint unsupervised learning of task segmentation and encoding; CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2022) and Diffusion-QL (Wang et al., 2023) are state-of-the-art offline RL methods, where Diffusion-QL uses the diffusion model as an expressive policy class. OPAL (Ajay et al., 2021), IRIS (Mandlekar et al., 2020), HiGoC (Li et al., 2022) and GoFAR (Ma et al., 2022) introduce a hierarchical structure in the offline goal-based RL to deal with the long horizon task; Diffuser (Janner et al., 2022) and DD (Ajay et al., 2022) are the same families of diffusion-based conditional generative models.

To increase the horizon, we splice trajectories in Maze2D and AntMaze randomly. Baselines are compared under single- and multi-task, as shown in Table 1 for the Maze2D dataset. The poor performance of MPPI shows the challenges of long-horizon tasks. The offline RL methods that introduce the hierarchical structure is better than flat algorithms, which verifies the rationality of our motivation. Diffusion-based conditional generative models exhibit the best performance, while HDMI shows a clear advantage in long-horizon goal-reaching tasks due to its hierarchical structure. In addition, the goal-based method still maintains performance in the multi-task setting without the noticeable performance drop exhibited by the flat methods.

In the AntMaze, the simplistic 2D ball from Maze2D is supplanted by the intricate 8-degrees-of-freedom "Ant" quadruped robot. The purpose of this numerical experiment is to rigorously assess the stitching challenge employing a morphologically sophisticated robot, closely emulating realistic navigation tasks in robotic systems. Table 2 shows

Table 1: The performance in Maze2D, a typical long-horizon task with reward sparsity. Multi2D is a multi-task variant with episodic, resampled goal locations. Results correspond to the mean and standard error over 5 planning seeds. The suffix number of the environment name indicates that the test map is stitched together from multiple original maps.

Environment		MPPI	CQL	IQL	OPAL	IRIS	HiGoC	Diffuser	DD	HDMI
Maze2D	U-Maze-3	14.4	3.6	23.2	-	63.8±2.5	61.2±3.3	82.6±1.6	83.9±3.1	**103.6±1.7**
Maze2D	Medium-2	5.7	2.3	19.8	-	59.5±4.7	59.8±4.1	87.8±3.1	85.8±3.3	**102.1±2.5**
Maze2D	Large-2	3.9	7.7	31.1	-	38.2±1.2	45.4±2.5	87.9±3.8	87.3±1.2	**104.7±2.1**
Single-task Average		8.0	6.8	24.7	-	53.8	55.5	86.1	85.7	**103.5**
Multi2D	U-Maze-3	17.8	-	16.5	-	61.7±3.6	67.9±1.5	85.4±1.8	86.9±3.5	**105.4±2.4**
Multi2D	Medium-2	8.1	-	8.9	62.3±2.8	41.4±1.9	52.4±3.7	85.6±3.4	88.2±1.3	**104.7±2.3**
Multi2D	Large-2	4.5	-	10.3	55.4±3.7	28.1±3.8	42.1±3.3	89.3±5.8	91.7±2.8	**105.8±1.9**
Multi-task Average		10.1	-	11.9	-	43.7	54.1	86.8	88.9	**105.3**

Table 2: The performance in AntMaze. MultiAnt-Diverse is a multi-task variant of AntMaze-Diverse with episodic, resampled goal locations. Results correspond to the mean and standard error over 5 planning seeds. The suffix number of the environment name indicates that the test map is stitched together from multiple original maps.

Environment		CompILE	GoFAR	DD	Diffusion-QL	HDMI
AntMaze-Play	U-Maze-3	41.2 ± 3.6	38.5 ± 2.2	73.1 ± 2.5	52.9 ± 4.1	**86.1± 2.4**
AntMaze-Diverse	U-Maze-3	23.5 ± 1.8	25.1 ± 3.1	49.2 ± 3.1	32.5 ± 5.9	**73.7±1.1**
AntMaze-Diverse	Large-2	-	-	46.8 ± 4.4	31.5 ± 4.5	**71.5±3.5**
Single-task Average		32.4	31.8	56.4	39.0	**77.1**
MultiAnt-Diverse	Large-2	-	-	45.2 ± 4.9	25.6 ± 5.8	**73.6 ± 3.8**
Multi-task Average		-	-	45.2	25.6	**73.6**

Table 3: The performance in D4RL, a standardize reward-maximizing environment, in terms of normalized average returns. Results for DD and HDMI correspond to the mean and standard error over 5 planning seeds.

Dataset	Environment	BC	CQL	IQL	DT	TT	MoReL	Diffuser	DD	Diffusion-QL	HDMI
Med-Expert	HalfCheetah	55.2	91.6	86.7	86.8	**95**	53.3	79.8	90.6±1.3	**96.8±0.3**	92.1±1.4
Med-Expert	Hopper	52.5	105.4	91.5	107.6	110.0	108.7	107.2	111.8±1.8	**111.1±1.3**	**113.5±0.9**
Med-Expert	Walker2d	**107.5**	**108.8**	**109.6**	**108.1**	101.9	95.6	**108.4**	**108.8±1.7**	**110.1±0.3**	**107.9±1.2**
Medium	HalfCheetah	42.6	44.0	47.4	42.6	46.9	42.1	44.2	**49.1±1.0**	**51.1±0.5**	**48.0±0.9**
Medium	Hopper	52.9	58.5	66.3	67.6	61.1	**95.4**	58.5	79.3±3.6	90.5±4.6	76.4±2.6
Medium	Walker2d	75.3	72.5	78.3	74.0	79	77.8	79.7	82.5±1.4	**87.0±0.9**	79.9±1.8
Med-Replay	HalfCheetah	36.6	**45.5**	**44.2**	36.6	41.9	40.2	42.2	39.3±4.1	**47.8±0.3**	**44.9±2.0**
Med-Replay	Hopper	18.1	95	94.7	82.7	91.5	93.6	96.8	**100±0.7**	**101.3±0.6**	**99.6±1.5**
Med-Replay	Walker2d	26.0	77.2	73.9	66.6	**82.6**	49.8	61.2	75±4.3	**95.5±1.5**	80.7±2.1
Average		51.9	77.6	77	74.7	78.9	72.9	75.3	81.8	**88.0**	82.6

Table 4: The performance of HDMI and baselines in FinRL, a realistic long-horizon reward-maximizing environment. Results are measured same as (Qin et al., 2022).

Dataset	Det.	BC	CQL	MB-PPO	DD	HDMI
FinRL-L-99	150	136	**487**	328	372	415
FinRL-L-999	150	137	416	656	721	**733**
FinRL-M-99	300	355	700	**1213**	830	1007
FinRL-M-999	300	504	621	698	712	**754**
FinRL-H-99	441	252	**671**	484	609	658
FinRL-H-999	441	270	444	787	782	**801**

the performance of different algorithms on the AntMaze dataset, and HDMI also shows a clear advantage.

4.2. Reward-Maximizing with Suboptimal Data

To verify the performance of HDMI on the reward-maximizing task and the ability to avoid suboptimal data pollution simultaneously, we selected the standardize benchmark, D4RL (Fu et al., 2020), for experiments. **Baselines:** In addition to some methods in §4.1, we also add baselines that perform well on the D4RL, including behavior cloning (BC); transformer-based conditional generative models DT (Chen et al., 2021) and TT (Janner et al., 2021); model-based RL method MoReL (Kidambi et al., 2020).

It can be seen from Table 3 that HDMI can outperform the specially designed offline RL algorithms on most tasks. While Diffusion-QL demonstrates superior performance

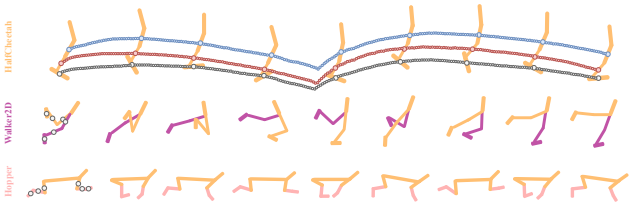


Figure 5: Visualization of partial subgoals and states sampled by HDMI in D4RL. Small circles indicate states, while large circles indicate subgoals. For the sake of clear presentation, we downsample the sequence and only visualized the state sequence sampled in the `Hopper` environment. The action sequences are not directly shown.

across all tasks on average, it falls short of HDMI on long-horizon tasks. It is important to acknowledge that despite the introduction of the conditional diffusion model, Diffusion-QL remains an offline RL method that substitutes the policy class with a more expressive, generative model. The performance gap between the diffusion- and transformer-based autoregressive models also verifies the superiority of the implicit planning effect brought about by synchronously generating the entire trajectory. Furthermore, the stability of HDMI also shows that planning-based subgoal extraction can effectively alleviate the pollution caused by suboptimal data. We visualized the subgoal and action sequence sampled by HDMI for different tasks, as shown in Figure 5.

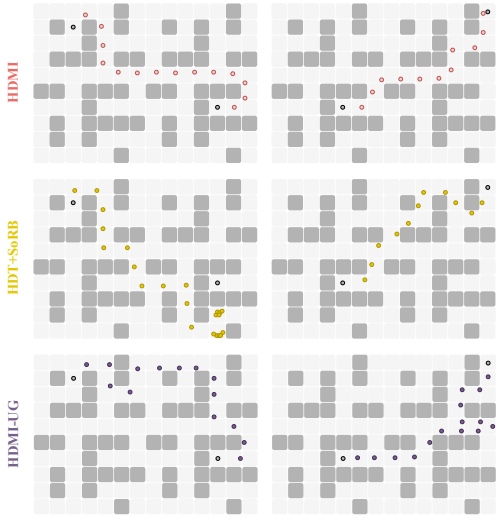


Figure 6: Subgoal sampled by baselines in `Large-2`.

4.3. Real-World Offline Decision-Making

Finally, considering the gap between D4RL and practical tasks in data distribution and evaluation indicators, we verified the practicability of HDMI on financial data in an industry benchmark, NeoRL (Qin et al., 2022). **Baselines:** In addition to some methods already introduced, we also introduce the MB-PPO (Qin et al., 2022) that performed well in NeoRL, where Det. means a deterministic working policy in the running system.

It can be seen from Table 4 that HDMI can stably match or even exceed the performance of the SOTA baselines. Another critical phenomenon is that DD and HDMI perform better on larger datasets, indicating that the diffusion model is prone to overfitting on small datasets.

4.4. Ablation Studies

This section verifies the advantages of the diffusion model synchronously generating the entire trajectory relative to the autoregressive model and transformer-based diffusion in modeling subgoal sequences (see §I.1 for more details). **Baselines.** The recently proposed hierarchical decision transformer, HDT (Correia & Alexandre, 2022); an ablation algorithm that replaces the target diffuser with U-Net (Janner et al., 2022), denoted as HDMI-UG.

We visualize the sampled subgoal in Figure 6. The subgoal sequence sampled by HDT tends to go around the long way, or even spins in a dead end. This may be because the autoregressive generation lacks the ability of implicit planning. HDMI-UG also shows a similar phenomenon. This may be due to the U-Net-based diffusion is more inclined to ensure local consistency and ignore the global constraints.

5. Closing Remarks

We propose a hierarchical diffusion probabilistic model with classifier-free guidance, HDMI, to solve the long-horizon offline decision-making problem. The upper-level return-based goal diffuser and the lower-level goal-based trajectory diffuser generate the optimal actions in a cascaded manner. Furthermore, HDMI adopts a planning-based subgoal extractor and the transformer-based diffusion to alleviate the suboptimal data pollution and the long-range subgoal dependence challenge. Numerical experiments validate the advantages of HDMI over SOTA offline RL and conditional generation models in handling long-horizon tasks, suboptimal data pollution, and realistic decision-making.

Acknowledgement

This work was supported in part by National Key Research and Development Program of China (No. 2020AAA0107400), STCSM (22QB1402100), NSFC (No. 12071145), Shenzhen Science and Technology Program (JCYJ20210324120011032), Postdoctoral Science Foundation of China (2022M723039), the Open Research Projects of Zhejiang Lab (NO. 2021KE0AB03), a grant from China Academy of LVT, and a grant from Shenzhen Institute of Artifcial Intelligence and Robotics for Society.

References

Agrawal, P., Carreira, J., and Malik, J. Learning to see by moving. In *ICCV*, 2015.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *ICLR*, 2021.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. Is conditional generative modeling all you need for decision-making? In *NeurIPS Foundation Models for Decision Making Workshop*, 2022.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. In *NeurIPS*, 2017.

Arthur, D. and Vassilvitskii, S. k-means++ the advantages of careful seeding. In *SODA*, 2007.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Dhruva, T., Muldal, A., Heess, N., and Lillicrap, T. Distributed distributional deterministic policy gradients. In *ICLR*, 2018.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *ICML*, 2017.

Camacho, E. F. and Alba, C. B. *Model predictive control*. Springer science & business media, 2013.

Carvalho, J., Baierl, M., Urain, J., and Peters, J. Conditioned score-based models for learning collision-free trajectory generation. In *NeurIPS Workshop on Score-Based Methods*, 2022.

Charpentier, A., Elie, R., and Remlinger, C. Reinforcement learning in economics and finance. *Computational Economics*, 2021.

Chebotar, Y., Hausman, K., Lu, Y., Xiao, T., Kalashnikov, D., Varley, J., Irpan, A., Eysenbach, B., Julian, R. C., Finn, C., et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In *ICML*, 2021.

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. In *NeurIPS*, 2021.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *NeurIPS*, 2018.

Correia, A. and Alexandre, L. A. Hierarchical decision transformer. *arXiv preprint arXiv:2209.10447*, 2022.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. In *ICML*, 2018.

De Lima, L. M. and Krohling, R. A. Discovering an aid policy to minimize student evasion using offline reinforcement learning. In *IJCNN*, 2021.

Ding, X., LI, Y.-t., and Chuan, S. Autonomic discovery of subgoals in hierarchical reinforcement learning. *The Journal of China Universities of Posts and Telecommunications*, 21(5):94–104, 2014.

Dorfman, R., Shenfeld, I., and Tamar, A. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. In *NeurIPS*, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. In *NeurIPS*, 2019.

Fatemi, M., Wu, M., Petch, J., Nelson, W., Connolly, S. J., Benz, A., Carnicelli, A., and Ghassemi, M. Semi-markov offline reinforcement learning for healthcare. In *CHIL*, 2022.

Floyd, R. W. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

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., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. In *ICLR*, 2022.

Ghosh, D., Ajay, A., Agrawal, P., and Levine, S. Offline rl policies should be trained to be adaptive. In *ICML*, 2022.

Grathwohl, W., Chen, R. T., Bettencourt, J., and Duvenaud, D. Scalable reversible generative models with free-form continuous dynamics. In *ICLR*, 2019.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T. L., Colmenarejo, S. G., Zolna, K., Agarwal, R., Merel, J., Mankowitz, D., Paduraru, C., Dulac-Arnold, G., Li, J., Norouzi, M., Hoffman, M., Nachum, O., Tucker, G., Heess, N., and deFreitas, N. Rl unplugged: Benchmarks for offline reinforcement learning. In *NeurIPS*, 2020.

Guo, X. and Zhai, Y. K-means clustering based reinforcement learning algorithm for automatic control in robots. *International Journal of Simulation Systems, Science & Technology*, 17(24):6.1–6.6, 2016.

Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent space policies for hierarchical reinforcement learning. In *ICML*, 2018.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, 2021.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

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

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. In *NeurIPS*, 2021.

Kipf, T., Li, Y., Dai, H., Zambaldi, V., Sanchez-Gonzalez, A., Grefenstette, E., Kohli, P., and Battaglia, P. CompILE: Compositional imitation learning and execution. In *ICML*, 2019.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *ICML*, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *ICLR*, 2022.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS*, 2016.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, 2020.

Kumar, A., Singh, A., Tian, S., Finn, C., and Levine, S. A workflow for offline model-free robotic reinforcement learning. In *CoRL*, 2022.

Lai, Y., Wang, W., Yang, Y., Zhu, J., and Kuang, M. Hindsight planner. In *AAMAS*, 2020.

Lee, H., Lu, J., and Tan, Y. Convergence for score-based generative modeling with polynomial complexity. In *NeurIPS*, 2022a.

Lee, T., Sung, D., Choi, K., Lee, C., Park, C., and Choi, K. Learning dynamic manipulation skills from haptic-play. *arXiv preprint arXiv:2207.14007*, 2022b.

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

Levine, S. and Koltun, V. Guided policy search. In *ICML*, 2013.

Li, J., Tang, C., Tomizuka, M., and Zhan, W. Hierarchical planning through goal-conditioned offline reinforcement learning. *arXiv preprint arXiv:2205.11790*, 2022.

Li, L., Munos, R., and Szepesvári, C. Toward minimax off-policy value estimation. In *AISTATS*, 2015.

Liu, J., Pan, F., and Luo, L. Gochat: Goal-oriented chatbots with hierarchical reinforcement learning. In *SIGIR*, 2020.

Liu, M., Zhu, M., and Zhang, W. Goal-conditioned reinforcement learning: Problems and solutions. In *IJCAI*, 2022.

Liu, Q., Lee, J., and Jordan, M. A kernelized stein discrepancy for goodness-of-fit tests. In *ICML*, 2016.

Liu, Q., Li, L., Tang, Z., and Zhou, D. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *NeurIPS*, 2018.

Liu, S. and Sun, S. Safe offline reinforcement learning through hierarchical policies. In *PAKDD*, 2022.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *CoRL*, 2020.

Ma, Y., Hao, X., Hao, J., Lu, J., Liu, X., Xialiang, T., Yuan, M., Li, Z., Tang, J., and Meng, Z. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. In *NeurIPS*, 2021.

Ma, Y. J., Yan, J., Jayaraman, D., and Bastani, O. Offline goal-conditioned reinforcement learning via f-advantage regression. In *NeurIPS*, 2022.

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 *ICRA*, 2020.

Mezghani, L., Sukhbaatar, S., Bojanowski, P., Lazaric, A., and Alahari, K. Learning goal-conditioned policies offline with self-supervised reward shaping. In *CoRL*, 2022.

Misra, D. Mish: A self regularized non-monotonic neural activation function. In *BMVC*, 2020.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. Nonparametric return distribution approximation for reinforcement learning. In *ICML*, 2010.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. Parametric return density estimation for reinforcement learning. *arXiv preprint arXiv:1203.3497*, 2012.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018.

Nachum, O., Chow, Y., Dai, B., and Li, L. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *NeurIPS*, 2019.

Nasiriany, S., Liu, H., and Zhu, Y. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *ICRA*, 2022.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

Parr, R. and Russell, S. Reinforcement learning with hierarchies of machines. In *NeurIPS*, 1997.

Pateria, S., Subagdja, B., and Tan, A. H. Hierarchical reinforcement learning with integrated discovery of salient subgoals. In *AAMAS*, 2020.

Pateria, S., Subagdja, B., Tan, A.-h., and Quek, C. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys*, 54(5):1–35, 2021.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Paul, S., Vanbaar, J., and Roy-Chowdhury, A. Learning from trajectories via subgoal discovery. In *NeurIPS*, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Pertsch, K., Lee, Y., Wu, Y., and Lim, J. J. Guided reinforcement learning with learned skills. In *CoRL*, 2021a.

Pertsch, K., Lee, Y., Wu, Y., and Lim, J. J. Demonstration-guided reinforcement learning with learned skills. In *CoRL*, 2021b.

Qin, R.-J., Zhang, X., Gao, S., Chen, X.-H., Li, Z., Zhang, W., and Yu, Y. NeoRL: A near real-world benchmark for offline reinforcement learning. In *NeurIPS Datasets and Benchmarks Track*, 2022.

Rao, D., Sadeghi, F., Hasenclever, L., Wulfmeier, M., Zambelli, M., Vezzani, G., Tirumala, D., Aytar, Y., Merel, J., Heess, N., et al. Learning transferable motor skills with hierarchical latent mixture policies. In *ICLR*, 2022.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-maron, G., Giménez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent. *Transactions on Machine Learning Research*, 11, 2022.

Ren, T., Li, J., Dai, B., Du, S. S., and Sanghavi, S. Nearly horizon-free offline reinforcement learning. In *NeurIPS*, 2021.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

Rosete-Beas, E., Mees, O., Kalweit, G., Boedecker, J., and Burgard, W. Latent plans for task-agnostic offline reinforcement learning. In *CoRL*, 2022.

Roy, B. Transitivité et connexité. *Comptes Rendus de l'Académie des Sciences*, 249:216–218, 1959.

Rudner, T. G., Pong, V., McAllister, R., Gal, Y., and Levine, S. Outcome-driven reinforcement learning via variational inference. In *NeurIPS*, 2021.

Salter, S., Wulfmeier, M., Tirumala, D., Heess, N., Riedmiller, M., Hadsell, R., and Rao, D. Mo2: Model-based offline options. In *CoLLAs*, 2022.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *ICML*, 2015.

Sculley, D. Web-scale k-means clustering. In *WWW*, 2010.

Seo, Y., Lee, K., James, S. L., and Abbeel, P. Reinforcement learning with action-free pre-training from videos. In *ICML*, 2022.

Shin, H. D. and Wang, R. E. Clap: Conditional latent planners for offline reinforcement learning. In *NeurIPS Foundation Models for Decision Making Workshop*, 2022.

Sinha, S., Mandlekar, A., and Garg, A. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *CoRL*, 2022.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Song, Y. Generative modeling by estimating gradients of the data distribution. *yang-song.net*, May 2021. URL https://yang-song.net/blog/2021/score/.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. In *ICML*, 2021.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Tang, S., Makar, M., Sjoding, M., Doshi-Velez, F., and Wiens, J. Leveraging factored action spaces for efficient offline reinforcement learning in healthcare. In *NeurIPS*, 2022.

Tedrake, R. *Underactuated Robotics*. 2022. URL https://underactuated.csail.mit.edu.

Uehara, M., Huang, J., and Jiang, N. Minimax weight and q-function learning for off-policy evaluation. In *ICML*, 2020.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. In *NeurIPS*, 2021.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *ICML*, 2017.

Villaflor, A. R., Huang, Z., Pande, S., Dolan, J. M., and Schneider, J. Addressing optimism bias in sequence modeling for reinforcement learning. In *ICML*, 2022.

Villecroze, V., Braviner, H., Naderian, P., Maddison, C., and Loaiza-Ganem, G. Bayesian nonparametrics for offline skill discovery. In *ICML*, 2022.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. In *ICLR*, 2023.

Warshall, S. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.

Williams, G., Drews, P., Goldfain, B., Rehg, J. M., and Theodorou, E. A. Aggressive driving with model predictive path integral control. In *ICRA*, 2016.

Wu, Y. and He, K. Group normalization. In *ECCV*, 2018.

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

Wulfmeier, M., Rao, D., Hafner, R., Lampe, T., Abdolmaleki, A., Hertweck, T., Neunert, M., Tirumala, D., Siegel, N., Heess, N., et al. Data-efficient hindsight off-policy option learning. In *ICML*, 2021.

Xie, R., Zhang, S., Wang, R., Xia, F., and Lin, L. Hierarchical reinforcement learning for integrated recommendation. In *AAAI*, 2021.

Xie, T., Ma, Y., and Wang, Y.-X. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. In *NeurIPS*, 2019.

Yang, M., Nachum, O., Dai, B., Li, L., and Schuurmans, D. Off-policy evaluation via the regularized lagrangian. In *NeurIPS*, 2020.

Yang, R., Lu, Y., Li, W., Sun, H., Fang, M., Du, Y., Li, X., Han, L., and Zhang, C. Rethinking goal-conditioned supervised learning and its connection to offline rl. In *ICLR*, 2022.

Yang, Y., Hu, H., Li, W., Li, S., Yang, J., Zhao, Q., and Zhang, C. Flow to control: Offline reinforcement learning with lossless primitive discovery. In *AAAI*, 2023.

Yu, T., Kumar, A., Chebotar, Y., Hausman, K., Levine, S., and Finn, C. Conservative data sharing for multi-task offline reinforcement learning. In *NeurIPS*, 2021.

Zhou, D., Wang, W., Yan, H., Lv, W., Zhu, Y., and Feng, J. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022.

Zhou, G., Azizsoltani, H., Ausin, M. S., Barnes, T., and Chi, M. Hierarchical reinforcement learning for pedagogical policy induction. In *IJCAI*, 2019.

Supplementary Material

A. Related Work

A.1. Hierarchical Framework in Offline RL

Many RL works focus on introducing a hierarchical framework for long-horizon offline decision-making by decomposing the task into a hierarchy of subproblems. This section will summarize the broader offline RL methods with a hierarchical framework rather than just migrating the HRL method to the offline setting. From the perspective of algorithm specificity, existing methods can be roughly divided into *goal*-based and *skill*-based (Pateria et al., 2021).

Goal-based methods mainly focus on solving a *single* task. Subproblems can be constructed by discovering their subgoals and then learning one subproblem policy to reach one subgoal or a set of related subgoals. A subgoal belongs to the environmental state space. The most crucial part of such methods lies in the selection or generation of subgoals. Once the upper-level subgoal is determined, the lower-level policy is generally learned using off-the-shelf offline RL methods based on the subgoal-augmented/conditioned policy or the universal value function (Schaul et al., 2015) or a combination of both. In existing methods, the subgoal is either predefined (Zhou et al., 2019; Xie et al., 2021; Ma et al., 2021); or selected based on heuristics, such as the frequency of states being visited by successful trajectories (Pateria et al., 2020), the manifold characteristics of adjacent states (Guo & Zhai, 2016), or the degree of restriction of associated actions (Ding et al., 2014); or generated by another offline RL supplemented with task-oriented intrinsic rewards (Liu et al., 2020; Liu & Sun, 2022).

Additionally, some methods are offline (i.e., learn from experience or expert trajectories) only during the subgoal selection or generation. (Eysenbach et al., 2019) build a graph on the replay buffer, taking the states as nodes and the distance from the start state to the goal state as the edge weights. Then use the graph search method to find the shortest path and take the nodes that make up the path as the subgoals. (Paul et al., 2019) uses the order of sub-goal indices along the expert trajectory as the evidence of difficulty and learns to generate the subgoals in the equipartitions from the expert trajectory conditioned on the given state. (Lai et al., 2020) selects subgoals uniformly in time or space from the expert trajectories.

Skill-based methods represent subgoals as low-dimensional latent vectors, i.e., "skills", which are not restricted to a subset of the state space. A skill refers to the policy of a subtask in the sense that it semantically represents *the ability to do something well* (Pateria et al., 2021). Therefore, skill-based methods can generally be used to solve *multi-task* problems. While these methods obtain a more powerful representation for subgoals, it also reduces the training efficiency due to the coupling of representation learning and reinforcement learning (Stooke et al., 2021; Seo et al., 2022). Naturally, it should be noted that not all methodologies oriented towards skill acquisition are necessarily grounded in the reinforcement learning framework. In the work presented by Kipf et al. (2019), denoted as CompILE, an alternative framework for learning hierarchically-structured behavior from offline datasets is proposed. Specifically, CompILE introduces an imitation learning approach that enables the acquisition of reusable, variable-length segments of skill. CompILE employs a novel unsupervised module for sequence segmentation that is fully differentiable, thereby allowing the extraction of latent encodings of sequential data. These encodings can subsequently be re-composed and executed in order to perform novel tasks.

More specifically, few works use predefined skills (Nasiriany et al., 2022; Fatemi et al., 2022). Lynch et al. (2020); Pertsch et al. (2021a); Ajay et al. (2021); Rosete-Beas et al. (2022); Lee et al. (2022b) learn skills with an auto-encoding loss function and incorporating a KL constraint to encourage better generalization. In addition to model-free methods, (Salter et al., 2022) builds on (Wulfmeier et al., 2021) with a model-based approach to generate skills from experience using dynamic programming. Unlike the above methods that can only discover a fixed number of skills, (Villecroze et al., 2022) dynamically adjusts the number of skills by introducing a Bayesian non-parametric method.

In addition, some works in *offline goal-conditioned reinforcement learning* (GCRL) also introduce hierarchical structures. Offline GCRL trains an agent to achieve *multi-tasks* under particular scenarios in the offline setting. They (Liu et al., 2022, Figure 1) either only focus on learning to master multiple goals simultaneously (Chebotar et al., 2021; Yang et al., 2022; Yu et al., 2021; Dorfman et al., 2021; Mezghani et al., 2022), or additionally decompose the long-horizon task into sub-goals (similar as goal-based methods) while learning to reach them with a unified policy (Li et al., 2022; Ma et al., 2022). This paper is interested in the latter. (Li et al., 2022) selects intermediate sub-goals by planning over future sub-goal sequences based on the learned value function of the low-level RL policy. (Ma et al., 2022) formulates the goal-reaching task as a state

occupancy matching problem between a dynamics-abiding imitator agent and an expert agent that directly teleports.

The proposed HDMI models the offline long-horizon decision-making as a conditional generation modeling problem based on the goal-based method, which allows HDMI to not only inherit the advantages of the goal-based method in terms of high training efficiency and interpretability but also apply to solving multi-task problems like the skill-based method.

A.2. Conditional Generative Models for Offline DM

Using TD-learning naively in offline decision-making will cause the state visitation distribution to move away from the dataset distribution. Offline RL resolves this distribution shift by imposing a constraint on the above distributions, but it also demands additional hyperparameter tuning and implementation heuristics (Kumar et al., 2022). Recent advancements in RL have sought to address the constraints of extant methodologies by employing conditional generative models as policy classes, which boast enhanced representational capacity. Wang et al. (2023) advocate for the implementation of a diffusion model as the policy representation, which is an emerging category of highly-expressive deep generative models. In this vein, the authors develop Diffusion-QL, a novel technique that employs a conditional diffusion model as the policy's representation. Diffusion-QL involves the acquisition of an action-value function and the incorporation of a term that maximizes action-values into the training loss of the conditional diffusion model. This integration results in a loss function that strives to identify optimal actions in close proximity to the behavior policy.

Instead of training an RL policy, recent works train conditional generation models using a sequence modeling objective. They do not face the risk of distribution shift as generative models are trained with maximum-likelihood estimation. (Chen et al., 2021) and (Janner et al., 2021) concurrently cast offline decision-making to return-conditioned generative modeling, and use transformers (Vaswani et al., 2017) to generate trajectories autoregressively. (Janner et al., 2022) introduces planning into the above-mentioned model-free style method and uses the diffusion model (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) to generate the optimal trajectory in a model predictive control (Camacho & Alba, 2013) (MPC) manner.

Unlike the above methods, which can only solve single tasks, follow-up work solves multi-task problems by extending the return-conditioned to the X-conditioned. For example, (Janner et al., 2021), (Shin & Wang, 2022), (Correia & Alexandre, 2022) and (Ajay et al., 2022) instantiate X as the goal or (predefined) skill; (Ajay et al., 2022) also instantiates X as the constraint to solve the multi-constraint problem. The proposed HDMI first formulates the problem of offline long-horizon decision-making from the perspective of conditional generative modeling. HDMI employs a hierarchical diffusion framework where the goal diffuser learns a reward-conditional diffusion model for the subgoal discovery, and the trajectory diffuser learns a goal-conditional diffusion model of the corresponding trajectory of subgoals.

B. Preliminaries

B.1. Offline Reinforcement Learning

We consider learning in a Markov decision process (MDP) described by the tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$. The MDP tuple consists of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $P\left(s'|s,a\right)$, and a reward function $r = \mathcal{R}(s,a)$. We use s_t, a_t, and $r_t = \mathcal{R}\left(s_t, a_t\right)$ to denote the state, action, and reward at timestep t, respectively. The return at timestep t, $R_t = \sum_{t'=t}^{\mathrm{T}} r_{t'}$, is the sum of future rewards from that timestep. The goal in (online) RL is to learn a policy that maximizes the expected return $\mathbb{E}\left[\sum_{t=1}^{\mathrm{T}} r_t\right]$ in an MDP by interacting with the natural environment or the simulator.

In the offline RL, we are given a static dataset of transitions $\mathcal{D} = \{(s_t, a_t, s_{t+1}, r_t)_i\}$, where i indexes a transition in the dataset, the actions come from the behavior policy $a_t \sim \pi_\beta\left(\cdot|s_t\right)$, the states come from a distribution induced by the behavior policy $s_t \sim d^{\pi_\beta}(\cdot)$, the next state is determined by the transition dynamics $s_{t+1} \sim P\left(\cdot|s_t, a_t\right)$, and the reward is still a function of state and action $r_t = E\left(s_t, a_t\right)$. The objective is the same as in the online case: to find a policy that maximizes the expected return. This setting is more problematic as it removes the ability to explore the environment.

B.2. Diffusion Models

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) are a likelihood-based generative model that learn the data distribution $q(\boldsymbol{\tau})$ from a fixed dataset $\mathcal{D} := \left\{\boldsymbol{\tau}^i\right\}$, where i indexes a sample in the dataset (Song, 2021). The critical idea of diffusion models is to model the (Stein) score function (Liu et al., 2016), which is not required to have a tractable normalizing constant. The discrete-time generating procedure is modeled with a hand-crafted forward noising process

$q\left(\boldsymbol{\tau}_{k+1}|\boldsymbol{\tau}_k\right) := \mathcal{N}\left(\boldsymbol{\tau}_{k+1}; \sqrt{\alpha_k}\boldsymbol{\tau}_k, \left(1-\alpha_k\right)\boldsymbol{I}\right)$ and a learnable reverse process $p_\theta\left(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k\right) := \mathcal{N}\left(\boldsymbol{\tau}_{k-1}|\mu_\theta\left(\boldsymbol{\tau}_k, k\right), \Sigma_k\right)$, where $\mathcal{N}(\mu, \Sigma)$ denotes a Gaussian distribution with mean μ and variance $\Sigma, \alpha_k \in \mathbb{R}$ determines the variance schedule, $\boldsymbol{\tau}_0 := \boldsymbol{\tau}$ is a sample, $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_{K-1}$ are the latents, and $\boldsymbol{\tau}_K \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for carefully chosen α_k and long enough K.

Starting with Gaussian noise, samples are then iteratively generated through a series of denoising steps. A tractable variational lower-bound on $\log p_\theta$ can be optimized to train the denosing process, (Ho et al., 2020) proposes a simplified surrogate loss:

$$\mathcal{L}_{\text{denoise}}\left(\theta\right) := \mathbb{E}_{k\sim[1,K], \boldsymbol{\tau}_0\sim q, \epsilon\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\epsilon - \epsilon_\theta\left(\boldsymbol{\tau}_k, k\right)\right\|^2\right].$$

The predicted noise $\epsilon_\theta\left(\boldsymbol{\tau}_k, k\right)$, parameterized with a deep neural network (e.g., U-Net (Ronneberger et al., 2015) or Transformers (Peebles & Xie, 2022)), approximates the noise $\epsilon \sim \mathcal{N}(0, I)$ added to the dataset sample $\boldsymbol{\tau}_0$ to produce noisy $\boldsymbol{\tau}_k$ in noising process.

Interestingly, the conditional distribution $q(\boldsymbol{\tau}|y(\boldsymbol{\tau}))$ makes it possible to generate samples with the condition $y(\boldsymbol{\tau})$. The equivalence between diffusion models and score-matching (Song et al., 2021), which shows $\epsilon_\theta\left(\boldsymbol{\tau}_k, k\right) \propto \nabla_{\boldsymbol{\tau}_k} \log p\left(\boldsymbol{\tau}_k\right)$, leads to two kinds of theoretically equivalent methods: classifier-guided (Nichol & Dhariwal, 2021), and classifier-free (Ho & Salimans, 2022) we are used. The later modifies the original training setup to learn both a conditional $\epsilon_\theta\left(\boldsymbol{\tau}_k, y(\boldsymbol{\tau}), k\right)$ and an unconditional $\epsilon_\theta\left(\boldsymbol{\tau}_k, k\right)$ model for the noise. The unconditional noise is represented as the conditional noise $\epsilon_\theta\left(\boldsymbol{\tau}_k, \varnothing, k\right)$ where a dummy value $\varnothing$ takes the place of $y(\boldsymbol{\tau})$. The perturbed noise $\epsilon_\theta\left(\boldsymbol{\tau}_k, k\right) + \omega\left(\epsilon_\theta\left(\boldsymbol{\tau}_k, y(\boldsymbol{\tau}), k\right) - \epsilon_\theta\left(\boldsymbol{\tau}_k, k\right)\right)$ is used to later generate samples.

B.3. Probabilistic Graphic Model for Control-as-Inference Framework

Based on the probabilistic graphical model depicted in Figure 2a and Haarnoja et al. (2018), our derivation in Section 2 is formulated. This model comprises factors for dynamics $p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$ and an action prior $p\left(\mathbf{a}_t\right)$, typically considered to be a uniform distribution. We introduce a binary random variable $\mathcal{O}_t$, referred to as the *optimality variable*, to each state and action because our interest lies in inferring the optimal trajectory distribution given a reward function.

To solve the optimal control problem, we can infer the posterior action distribution $\pi^*\left(\mathbf{a}_t \mid \mathbf{s}_t\right) = p\left(\mathbf{a}_t \mid \mathbf{s}_t, \mathcal{O}_{t:T} = \text{true}\right)$. It implies that an optimal action is one where the optimality variable is active for the current state and all subsequent states. To maintain brevity in our derivation, we will refrain from explicitly writing $\mathcal{O}_t = \text{true}$ but instead denote $\mathcal{O}_t$ to represent the state-action tuple corresponding to the time at which it was optimal.

To incorporate the reward function into this framework, we can select $p\left(\mathcal{O}_t \mid \mathbf{s}_t, \mathbf{a}_t\right) = \exp\left(r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)$, which assumes, without loss of generality, that $r\left(\mathbf{s}_t, \mathbf{a}_t\right) < 0$. We can express the distribution over optimal trajectories as

$$p\left(\tau \mid \mathcal{O}_{0:T}\right) \propto p\left(\mathbf{s}_0\right) \prod_{t=0}^{T} p\left(\mathbf{a}_t\right) p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right) \exp\left(r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)$$

This distribution can be utilized to make various queries, such as $p\left(\mathbf{a}_t \mid \mathbf{s}_t, \mathcal{O}_{t:T}\right)$.

C. Missing Proofs

C.1. Proof of Proposition 2.2

Proof.

$$\begin{aligned} p\left(\boldsymbol{\tau}_0 \mid y(\boldsymbol{\tau}_0)\right) &= p\left(\boldsymbol{\tau}_0 \mid \Upsilon_{0:N-1} = 1\right) \propto p\left(\boldsymbol{\tau}_0\right) p\left(\Upsilon_{0:N-1} = 1 \mid \boldsymbol{\tau}_0\right) \\ &= p(s_0) \prod_{i=1}^{N-1} p(g_i) p(\Upsilon_i \mid g_i, \{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M}) \\ &\quad \prod_{j=1}^{M} p(a_{(i-1)*M+j} \mid s_{(i-1)*M+j}, g_i) p(s_{(i-1)*M+j+1} \mid s_{(i-1)*M+j}, a_{(i-1)*M+j}) \end{aligned}$$

$$
\begin{aligned}
&\propto p(s_0) \prod_{i=1}^{N-1} p(g_i) \exp\left(\sum_{j=1}^{M} r\left(s_{(i-1)*M+j}, a_{(i-1)*M+j}\right)\right) \\
&\quad \prod_{j=1}^{M} p(a_{(i-1)*M+j} \mid s_{(i-1)*M+j}, g_i) p(s_{(i-1)*M+j+1} \mid s_{(i-1)*M+j}, a_{(i-1)*M+j}) \\
&= \overbrace{\exp\left(\sum_{i=1}^{N-1}\sum_{j=1}^{M} r\left(s_{(i-1)*M+j}, a_{(i-1)*M+j}\right)\right)}^{y(\tau_g)} \overbrace{p(s_0) \prod_{i=1}^{N-1} p(g_i)}^{p(\tau_g)} \\
&\quad \underbrace{\prod_{j=1}^{M} p(a_{(i-1)*M+j} \mid s_{(i-1)*M+j}, g_i) p(s_{(i-1)*M+j+1} \mid s_{(i-1)*M+j}, a_{(i-1)*M+j})}_{p(\tau_{sa}^i) y(\tau_{sa}^i)} \\
&= p(\tau_g) y(\tau_g) \prod_{i=1}^{N} p(\tau_{sa}^i) y(\tau_{sa}^i),
\end{aligned}
$$

where $s_0 \equiv g_0$, τ_g denotes subgoal sequence $g_{0:N-1}$, τ_{sa}^i represents the trajectory $\{s_{(i-1)*M+j}, a_{(i-1)*M+j}\}_{j=1}^{M})$ corresponding to the subgoal g_i, and $y(\tau_{sa}^i)$ is a Dirac delta for observed values and constant elsewhere. Concretely, if subgoal g_i is state constraint at timestep $i * M$, then

$$
y(\tau_{sa}^i) = \delta_{g_i}\left(s_{(i-1)*M+1}, a_{(i-1)*M+1}, \ldots, s_{i*M}, a_{i*M}\right) = \begin{cases} +\infty, \text{ if } \mathbf{g}_i = s_{i*M}, \\ 0, \text{ otherwise }. \end{cases} \tag{5}
$$

□

D. Reward-Maximizing Decision-Making

Algorithm 2 and Algorithm 3 are the model training and the optimal action sequence sampling pseudocode (for the testing phase) of HDMI[3], respectively, in the reward-maximizing decison-making task. It is worth noting that Algorithm 2 does not distinguish between the goal and trajectory diffuser, and the subgoal sequence $\boldsymbol{\tau}$ and the state sequence $\boldsymbol{\tau}_{sa}$ are represented using the uniform symbol $\boldsymbol{\tau}$.

D.1. Model Training

Algorithm 2 Reward-Maximizing Diffusion Model Training with Classifier-free Guidance.

Input: the offline dataset $\mathcal{D}$, the probability of unconditional training p_u (fixed $p_u = 1$ for the trajectory diffuser).
repeat
  $(\boldsymbol{\tau}_0, y(\boldsymbol{\tau}_0)) \sim \mathcal{D}$ ▷ Sample subgoal/state trajectory with conditioning from the dataset
  $y(\boldsymbol{\tau}_0) \leftarrow \varnothing$ with probability p_{u} ▷ Randomly discard conditioning to train unconditionally
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), k \sim \mathcal{U}\{1, \ldots, K\}$
  $\boldsymbol{\tau}_{g,k} := \sqrt{\alpha_k}\boldsymbol{\tau}_{g,0} + (1-\alpha_k)\epsilon$ ▷ Corrupt data to the sampled value
  Take gradient step on $\nabla_\theta \left\|\epsilon - \hat{\epsilon}_\theta\right\|^2$ where $\hat{\epsilon}_\theta := \epsilon_\theta\left(\boldsymbol{\tau}_k, y(\boldsymbol{\tau}_k), k\right)$ ▷ Optimization of denoising model
until converged
Output: the parameter θ of the diffusion model.

During the training phase, for each sampled subgoal trajectory $\boldsymbol{\tau}_{g,0}$, we first sample noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a timestep $k \sim \mathcal{U}\{1, \ldots, K\}$. Then, we construct a noised subgoal sequence $\boldsymbol{\tau}_{g,k} := \sqrt{\alpha_k}\boldsymbol{\tau}_{g,0} + (1-\alpha_k)\epsilon$. Finally, we predict the

[3]The code is available at https://anonymous.4open.science/r/HDMI/.

noise as $\hat{\epsilon}_{\theta_g} := \epsilon_{\theta_g}\left(\boldsymbol{\tau}_{g,k}, (1-\beta)y(\boldsymbol{\tau}_{g,k}) + \beta\varnothing, k\right)$, with probability p_u we ignore the conditioning information.

The trajectory diffuser is trained similarly to the goal diffuser, but we need to make a few more trade-offs. Firstly, we only use the trajectory diffuser to estimate the posterior distribution of state trajectories $\boldsymbol{\tau}^i_{s,0} := \{s_{(i-1)*M+j}\}_{j=1}^{M}$, and use inverse dynamics model (Agrawal et al., 2015; Pathak et al., 2017) similarly to (Ajay et al., 2022) to generate action trajectories based on state trajectories.

Secondly, the conditioning information $y(\boldsymbol{\tau}^i_{sa,0}))$ of the trajectory diffuser is a Dirac delta for the subgoal constraints as mentioned in Proposition 2.2. We fed these constrains into the trajectory diffuser by sampling from the denoising process $\boldsymbol{\tau}^i_{s,k-1} \sim p_{\theta_s}\left(\boldsymbol{\tau}^i_{s,k-1} | \boldsymbol{\tau}^i_{s,k}\right)$ and replacing the last state of the sampled trajectory with conditioning subgoal g_i after each reverse timesteps k. This means that when training the trajectory diffuser, we do not need to input the conditioning information $y(\boldsymbol{\tau}^i_{sa,0}))$ but only need to use the above trick when generating trajectories during the test phase.

D.2. Model Sampling

Algorithm 3 Reward-Maximizing Decision-Making with the Hierarchical Diffusion.

Input: goal diffuser ϵ_{θ_g}, trajectory diffuser ϵ_{θ_s}, inverse dynamics model F_{θ_I}, classifier-free guidance scale w, starting subgoal $g_0 \leftarrow s_0$, fixed conditioning information $y(\boldsymbol{\tau}_g) \leftarrow 1$, initializing subgoal history $h_g.\texttt{insert}(g_0)$.
while not done **do** ▷ Goal diffusion
  initialize $\boldsymbol{\tau}_g \sim \mathcal{N}(0, \alpha I)$ ▷ Sample noise subgoal trajectory
  for $k = K_g, \ldots, 1$ **do** ▷ Receding horizon control loop
    $\boldsymbol{\tau}_{g,k}[:\texttt{length}(h_g)] \leftarrow h_g$ ▷ Constrain newly generated subgoals are consistent with already generated subgoals
    $\hat{\epsilon} \leftarrow \epsilon_{\theta_g}\left(\boldsymbol{\tau}_{g,k}, \varnothing, k\right) + \omega\left(\epsilon_{\theta_g}\left(\boldsymbol{\tau}_{g,k}, y(\boldsymbol{\tau}_g), k\right) - \epsilon_{\theta_g}\left(\boldsymbol{\tau}_{g,k}, \varnothing, k\right)\right)$ ▷ Classifier-free guidance
    $\left(\mu_{k-1}, \Sigma_{k-1}\right) \leftarrow \texttt{Denoise}\left(\boldsymbol{\tau}_{g,k}, \hat{\epsilon}\right)$
    $\boldsymbol{\tau}_{g,k-1} \sim \mathcal{N}\left(\mu_{k-1}, \alpha\Sigma_{k-1}\right)$
  end for
  observe next subgoal g; $h_g.\texttt{insert}(g)$
  initialize state history h_s, $t \leftarrow 0$
  while not done **do** ▷ Trajectory diffusion
    observe state s; $h_s.\texttt{insert}(s)$; initialize $\boldsymbol{\tau}_s \sim \mathcal{N}(0, \alpha I)$ ▷ Sample noise state trajectory
    for $k = K_g, \ldots, 1$ **do** ▷ Receding horizon control loop
      $\boldsymbol{\tau}_{s,k}[:\texttt{length}(h_s)] \leftarrow h_s$ ▷ Constrain newly generated states are consistent with already generated states
      $\hat{\epsilon} \leftarrow \epsilon_{\theta_s}\left(\boldsymbol{\tau}_{s,k}, \varnothing, k\right)$ ▷ Unconditional diffusion
      $\left(\mu_{k-1}, \Sigma_{k-1}\right) \leftarrow \texttt{Denoise}\left(\boldsymbol{\tau}_{s,k}, \hat{\epsilon}\right)$
      $\boldsymbol{\tau}_{s,k-1} \sim \mathcal{N}\left(\mu_{k-1}, \alpha\Sigma_{k-1}\right)$
      $\boldsymbol{\tau}_{s,k-1}[-1] \leftarrow g$ ▷ Reformulating the conditional generation to the inpainting problem
    end for
    Extract $\left(s_t, s_{t+1}\right)$ from $\boldsymbol{\tau}_{s,0}$
    Execute $a_t = F\left(s_t, s_{t+1}; \theta_I\right); t \leftarrow t+1$
  end while
end while
Output: the optimal action sequence $\{a_t\}_{t=0}^{T-1}$ of the decision-making problem.

After the model is trained, sampling from the goal and the trajectory diffuser is equivalent to complete the planning in RL to solve the decision-making task. Concretely, given a decision-making task, to sample an optimal trajectory, we first conditionally sample the optimal subgoal sequence τ_g based on the information $y(\tau_g)$; and then conditionally samples the optimal trajectory τ^i_{sa} corresponding to the subgoal g_i based on the information $y(\tau^i_{sa})$. The details are clearly shown in Algorithm 3.

There are two points need to be explained: the classifier-free guidance and the receding horizon control (RHC). For the former we use a discrete-time version of (Ho & Salimans, 2022). And for the latter, due to the aleatoric uncertainty of the environment and the epistemic uncertainty of the model, directly generating the subgoal and the corresponding action sequence in the test phase will lead to errors that are exponentially proportional to the planning horizon.

Thus, we adopt a RHC method subordinate to MPC similar to (Ajay et al., 2022). RHC does not perform fixed horizon

optimization like MPC. The horizon at each sampling step is one less than the previous step, but it relies on one more step information than the previous step, as shown in Algorithm 3. The RHC method is able to take full advantage of the transformer-based diffusion model in modeling long-range dependencies. It is worth noting that to utilize the already sampled trajectory information, we employ a similar inpainting trick to that used in the trajectory diffuser.

E. Goal-Reaching Decision-Making

The model training and sampling so far is mainly aimed at the reward-maximizing task. That is, the conditional information based on the generation of the goal trajectory is related to the return. However, HDMI is equally applicable to solving the goal-reaching problem (i.e., the agent is only rewarded when it approaches the goal state), simply by applying the training and sampling method of the trajectory diffuser to the goal diffuser.

Algorithm 4 Goal-Reaching Diffusion Model Training with Classifier-free Guidance.

Input: the offline dataset $\mathcal{D}$
repeat
  $\boldsymbol{\tau}_0 \sim \mathcal{D}$ ▷ Sample subgoal/state trajectory from the dataset
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), k \sim \mathcal{U}\{1, \ldots, K\}$
  $\boldsymbol{\tau}_{g,k} := \sqrt{\alpha_k}\boldsymbol{\tau}_{g,0} + (1-\alpha_k)\epsilon$ ▷ Corrupt data to the sampled value
  Take gradient step on $\nabla_\theta \left\|\epsilon - \hat{\epsilon}_\theta\right\|^2$ where $\hat{\epsilon}_\theta := \epsilon_\theta\left(\boldsymbol{\tau}_k, \varnothing, k\right)$ ▷ Optimization of denoising model
until converged
Output: the parameter θ of the diffusion model.

Specifically, in the training phase, as shown in Algorithm 4, the target diffuser is also no longer based on conditional information, but is trained as an unconditional generative model.

And in the testing phase, in order to sample the optimal action trajectory that satisfies the goal constraints, we only need to first use the unconditional goal or trajectory diffuser to sample the sequence of subgoals or states before using the inpainting trick to replace the last subgoal or state with the goals corresponding to the constraints, as shown in Algorithm 5.

F. Missing Results and Graphs

Table 5: The performance of HDMI and baselines in Maze2D, a typical long-horizon task with reward sparsity. The Multi2D setting refers to a multi-task variant with episodic, resampled goal locations. Results correspond to the mean and standard error over 5 planning seeds. We emphasize in bold scores within 5 percent of the maximum per task (Kostrikov et al., 2022).

Environment		MPPI	CQL	IQL	OPAL	IRIS	HiGoC	Diffuser	DD	HDMI
Maze2D	U-Maze	33.2	5.7	47.4	-	82.6±4.7	85.3±2.1	113.9±3.1	116.2±2.7	**120.1±2.5**
Maze2D	Medium	10.2	5.0	34.9	-	73.1±4.5	81.4±2.4	121.5±2.7	**122.3±2.1**	121.8±1.6
Maze2D	Large	5.1	12.5	58.6	-	57.9±3.6	69.1±2.3	123.0±6.4	125.9±1.6	**128.6±2.9**
Single-task Average		16.2	7.7	47.0	-	71.2	78.6	119.5	121.5	**123.5**
Multi2D	U-Maze	41.2	-	24.8	-	89.4±2.4	91.2±1.9	128.9±1.8	128.2±2.1	**131.3±1.8**
Multi2D	Medium	15.4	-	12.1	81.1±3.1	64.8±2.6	79.3±2.5	127.2±3.4	129.7±2.7	**131.6±1.9**
Multi2D	Large	8.0	-	13.9	70.3±2.9	43.7±1.3	67.3±3.1	132.1±5.8	130.5±4.2	**135.4±2.5**
Multi-task Average		21.5	-	16.9	-	66.0	79.3	129.4	129.5	**132.8**

In the main part to verify the advantages of HDMI over baselines under long-horizon tasks, we stitch the original Maze2D dataset. Figure 7 shows visually how the stitching of the trajectory is done. In order to smooth the trajectory, we sometimes need to flip the map. For the endpoints of different trajectories, we will connect them by a dash. In this section we post the performance of HDMI on the original Maze2D dataset, as shown in Table 5. The algorithms exhibit similar characteristics to those analyzed in the experimental section of the main text. A point worth noting here is that the performance of Diffuser as well as DD is closer to that of HDMI in Table 5, but there is a large difference in Table 1. This again verifies the reasonableness and effectiveness of the hierarchical structure introduced by HDMI.

We also visualized the subgoal and action sequence sampled by HDMI for different tasks , as shown in Figure 8 and 9.

Algorithm 5 Goal-Reaching Decision-Making with the Hierarchical Diffusion.

Input: goal diffuser ϵ_{θ_g}, trajectory diffuser ϵ_{θ_s}, inverse dynamics model F_{θ_I}, classifier-free guidance scale w, starting subgoal $g_0 \leftarrow s_0$, goal state s_T, initializing subgoal history $h_g.\texttt{insert}(g_0)$.
while not done **do** ▷ Goal diffusion
  initialize $\boldsymbol{\tau}_g \sim \mathcal{N}(0, \alpha I)$ ▷ Sample noise subgoal trajectory
  for $k = K_g, \ldots, 1$ **do** ▷ Receding horizon control loop
    $\boldsymbol{\tau}_{g,k}[: \texttt{length}(h_g)] \leftarrow h_g$ ▷ Constrain newly generated subgoals are consistent with already generated subgoals
    $\hat{\epsilon} \leftarrow \epsilon_{\theta_g}(\boldsymbol{\tau}_{g,k}, \varnothing, k)$ ▷ Unconditional diffusion
    $(\mu_{k-1}, \Sigma_{k-1}) \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_{g,k}, \hat{\epsilon})$
    $\boldsymbol{\tau}_{g,k-1} \sim \mathcal{N}(\mu_{k-1}, \alpha\Sigma_{k-1})$
    $\boldsymbol{\tau}_{g,k-1}[-1] \leftarrow s_T$ ▷ Reformulating the conditional generation to the inpainting problem
  end for
  observe next subgoal g; $h_g.\texttt{insert}(g)$
  initialize state history h_s, $t \leftarrow 0$
  while not done **do** ▷ Trajectory diffusion
    observe state s; $h_s.\texttt{insert}(s)$; initialize $\boldsymbol{\tau}_s \sim \mathcal{N}(0, \alpha I)$ ▷ Sample noise state trajectory
    for $k = K_g, \ldots, 1$ **do** ▷ Receding horizon control loop
      $\boldsymbol{\tau}_{s,k}[: \texttt{length}(h_s)] \leftarrow h_s$ ▷ Constrain newly generated states are consistent with already generated states
      $\hat{\epsilon} \leftarrow \epsilon_{\theta_s}(\boldsymbol{\tau}_{s,k}, \varnothing, k)$ ▷ Unconditional diffusion
      $(\mu_{k-1}, \Sigma_{k-1}) \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_{s,k}, \hat{\epsilon})$
      $\boldsymbol{\tau}_{s,k-1} \sim \mathcal{N}(\mu_{k-1}, \alpha\Sigma_{k-1})$
      $\boldsymbol{\tau}_{s,k-1}[-1] \leftarrow g$ ▷ Reformulating the conditional generation to the inpainting problem
    end for
    Extract (s_t, s_{t+1}) from $\boldsymbol{\tau}_{s,0}$
    Execute $a_t = F(s_t, s_{t+1}; \theta_I)$; $t \leftarrow t + 1$
  end while
end while
Output: the optimal action sequence $\{a_t\}_{t=0}^{T-1}$ of the decision-making problem.

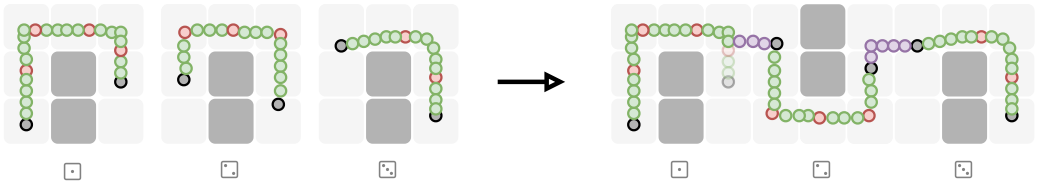


Figure 7: Construction of U-Maze-3 test scenarios and stitching of trajectories in the training set. Dark gray circles indicate starting or ending points. Red and green circles denote subgoals and states, respectively. The purple circles represent the state of replenishment. For the sake of clear presentation, we downsample the sequence.

In order to further substantiate the efficacy of the HDMI approach in addressing more intricate *sub-goal-based* reward-maximizing problems, we have incorporated the FrankaKitchen dataset from the D4RL in our analysis. The FrankaKitchen entails the manipulation of a 9-DoF Franka robotic system within a kitchen setting that features a variety of commonplace household objects, including a microwave, kettle, overhead light, cabinets, and an oven. The primary objective of each task is to engage with these items to achieve a specified goal configuration. More specifically, we have opted for the "mixed" datasets, which comprise undirected data wherein the robotic system carries out subtasks that do not necessarily correlate with the target configuration. Notably, this dataset lacks any trajectories that entirely resolve the task; thus, the RL agent must acquire the ability to amalgamate the pertinent sub-trajectories. Our examination of Table 6 leads us to corroborate the conclusions drawn from the aforementioned analysis.

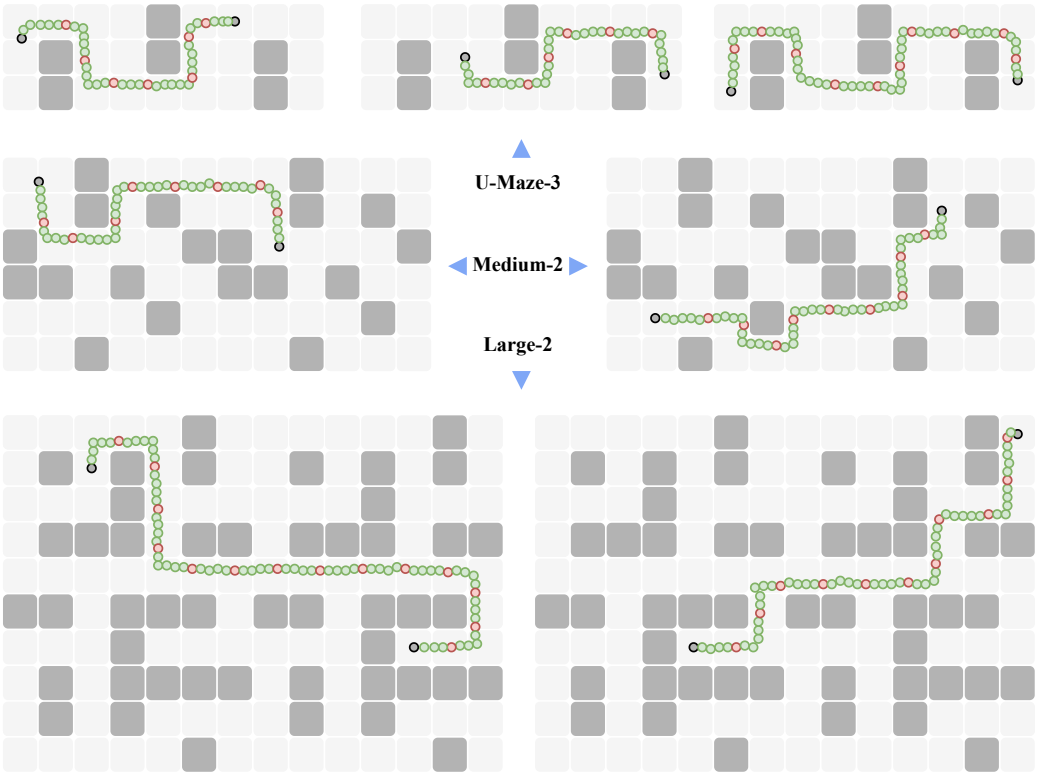


Figure 8: Visualization of subgoal and action sequence sampled by HDMI in long-horizon goal-reaching decision-making. Dark gray circles indicate starting or ending points. Red and green circles denote subgoals and states, respectively. The action sequences are not directly shown in the figure. For the sake of clear presentation, we downsample the sequence.

Table 6: The performance in Kitchen, a standardize goal-based reward-maximizing environment, in terms of normalized average returns. "Mixed" datasets contain no trajectories which solve the task completely, and the RL agent must learn to assemble the relevant sub-trajectories. Results correspond to the mean and standard error over 5 planning seeds.

Environment		CompILE	GoFAR	DD	Diffusion-QL	HDMI
Mixed	Kitchen	52.3 ± 1.9	44.5 ± 2.3	65.0 ± 2.8	62.6 ± 5.1	$\mathbf{69.2 \pm 1.8}$

G. Missing Discussions

#Q1: Whether HDMI does better on long-horizon tasks due to its improved architecture design? It is challenging to compare HDMI with other baselines directly on network architecture due to fundamental differences in multiple dimensions, such as whether the algorithms belong to RL or generative models, whether the generative models are based on autoregressive models or diffusion models, etc. Therefore, the following analysis will be presented in three parts: 1) a comparison among offline RL algorithms; 2) a comparison among generative methods, and 3) a comparison between offline RL and generative methods. These comparisons lead to three main conclusions.

In offline reinforcement learning, diffusion models have significant advantages when the network structure is similar. Diffusion-QL belongs to the offline RL algorithm, and its network structure is similar to other RL baselines, mainly based on MLP or CNN. However, the performance comparison between diffusion models and other RL baselines indicates that diffusion models have stronger representation capabilities than ordinary policy classes, which leads to a significant performance advantage for Diffusion-QL.

Among generative methods, diffusion models also have significant advantages. HDMI is more proficient in handling long-horizon tasks by combining hierarchical architectures and transformers. The generative baselines selected in this paper are mainly divided into two categories: one is based on autoregressive models, such as DT, TT, mainly based on the transformer network structure; the other is based on diffusion models, such as Diffuser, DD, mainly based on the

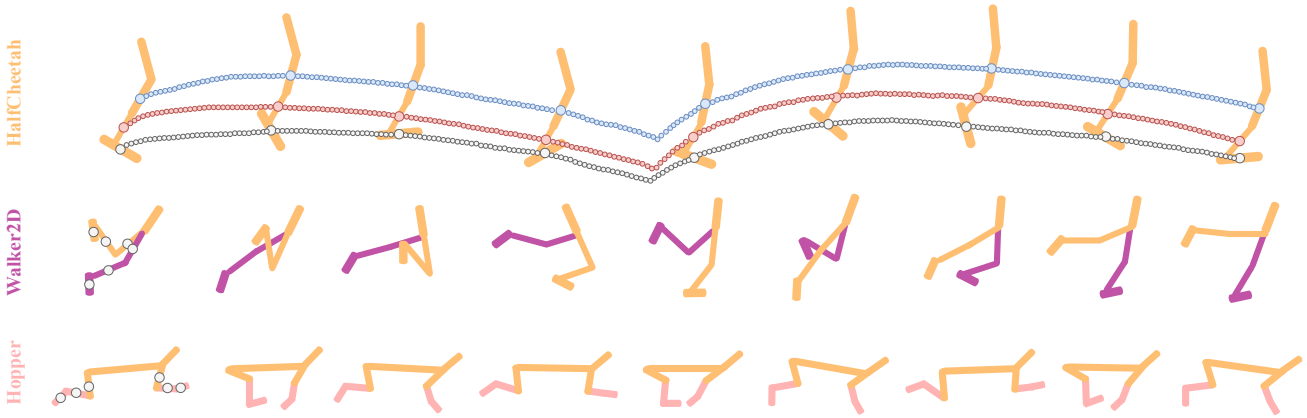


Figure 9: Visualization of partial subgoal and state sequence sampled by HDMI in long-horizon reward-maximizing decision-making. Small circles of different colors indicate state sequences, while large circles indicate subgoals. For the sake of clear presentation, we downsample the sequence and only visualized the state sequence sampled in the `Hopper` environment. The action sequences are not directly shown in the figure.

U-Net network structure. From the experimental results, it can be observed that diffusion models based on U-Net are more advantageous. To handle long-horizon tasks, we made improvements to the architecture of the diffusion model by introducing a hierarchical architecture and replacing U-Net with transformers. Performance comparisons and ablation experiments show that the hierarchical architecture and the diffusion model based on transformers are more advantageous in handling long-horizon tasks.

In handling long-horizon tasks, hierarchical architectures are more advantageous, and generative methods outperform RL methods. Lastly, we compared the RL algorithms with generative methods. In the reward-maximizing D4RL with suboptimal data contamination, there is little difference in the optimal performance between the two types of methods, with the RL method based on diffusion models slightly superior. In the long-horizon goal-reaching task, algorithms with hierarchical architectures in both categories outperform others, and generative methods are more advantageous than RL.

#Q2: What is the connection between Rudner et al. (2021) and HDMI? For convenience of analysis, we first paste the necessary formulas below, starting with the Equation 2:

$$p\left(\boldsymbol{\tau}_0 \mid y(\boldsymbol{\tau}_0)\right) \propto p(\tau_g) y(\tau_g) \prod_{i=1}^{N} p(\tau_{sa}^i) y(\tau_{sa}^i),$$

and the Equation (5) Rudner et al. (2021):

$$p_{\tilde{\mathcal{T}}_{0:T}, \mathbf{s}_{T^\star}, T \mid \mathbf{s}_0}\left(\tilde{\boldsymbol{\tau}}_{0:t}, \mathbf{g}, t \mid \mathbf{s}_0\right) = p_T(t) p_d\left(\mathbf{g} \mid \mathbf{s}_t, \mathbf{a}_t\right) p\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \prod_{t'=0}^{t-1} p_d\left(\mathbf{s}_{t'+1} \mid \mathbf{s}_{t'}, \mathbf{a}_{t'}\right) p\left(\mathbf{a}_{t'} \mid \mathbf{s}_{t'}\right).$$

Although there are significant differences in their forms, we find that the probability model of each sequence corresponding to the sub-goal in the first formula, i.e., $p(\tau_{sa}^i) y(\tau_{sa}^i)$, has the same physical meaning as the probability model of the sequence in the second formula. Specifically, $y(\tau_{sa}^i)$ represents that the final state of the sequence τ_{sa}^i reaches the goal state g_i. The second formula also represents that the final state of the sequence reaches a specified goal state g. However, since this paper does not use variational inference to learn the goal-conditioned policy, $y(\tau_{sa}^i)$ is not expanded into the form of $p_T(t) p_d\left(\mathbf{g} \mid \mathbf{s}_t, \mathbf{a}_t\right)$ in the second formula.

In other words, to solve the long-horizon decision-making problem, Section 2 introduces the goal in the control-as-inference framework, forming a hierarchical structure. In this hierarchical structure, the upper layer, which is the probability model of the goal sequence, is equivalent to the original control-as-inference framework. The lower layer, which is the probability model of several sub-sequences corresponding to the subgoal, is equivalent to the outcome-driven reinforcement learning framework proposed by Rudner et al. (2021).

In general, the hierarchical framework proposed in Section 2 integrates the control-as-inference framework and the outcome-driven reinforcement learning framework proposed by Rudner et al. (2021). After converting the hierarchical architecture

into a hierarchical conditional generation problem and combining it with the hierarchical diffusion model, the HDMI algorithm can effectively handle offline long-horizon decision-making problems, which cannot be achieved by either the control-as-inference framework or the outcome-driven reinforcement learning framework alone. Meanwhile, Rudner et al. (2021) mainly designed for goal-reaching tasks and thus are not proficient in solving reward-maximizing tasks. However, due to the upper layer control-as-inference framework, HDMI has good versatility and can handle both goal-reaching and reward-maximizing tasks.

H. Hyperparameter and Training Details

This section will give details of the hyperparameter settings and training details in numerical experiments of the baselines as well as the proposed HDMI. For performances of baselines previously evaluated on standardized tasks, we provide the source of the listed performances. We use the same settings as (Janner et al., 2022) and (Li et al., 2022) for some baselines.

H.1. Baseline Details

H.1.1. Goal-Reaching (Single-Task)

- The performance of CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2022) in Table 5 is reported in the D4RL (Fu et al., 2020, Table 2);
- The performance of OPAL (Ajay et al., 2021), IRIS (Mandlekar et al., 2020) and HiGoC (Li et al., 2022) in Table 5 is reported in the HiGoC (Li et al., 2022, Table III);
- The performance of Diffuser (Janner et al., 2022) in Table 5 is reported in the (Janner et al., 2022, Table 1).

We run DD using the offical repository[4] from the original paper with default hyperparameters.

H.1.2. Goal-Reaching (Multi-Task)

For offline RL baselines, we only evaluated IQL on the multi-task setting because it is the strongest baseline in the single-task goal-reaching by a sizeable margin. The performance of all baselines in Table 5 are from the same source as the single-task setting.

H.1.3. Long-Horizon Goal-Reaching (Single-Task)

- We run CQL using the offical repository[5] from the original paper, and tune over the two hyparameters, Q-function learning rate $\in [1e-4, 3e-4]$ and Lagrange threshold $\in [2.0, 10.0]$;
- We run IQL using the offical repository[6] from the original paper, and tune over the two hyparameters, temperature $\in [3, 10]$ and expectile $\in [0.65, 0.95]$, same as (Janner et al., 2022);

For other baselines that require trajectories as input, we did not modify the slice lengths in these methods, thus eliminating the need for model resizing.

- We re-implement IRIS based on BCQ[7] and cVAE[8] with default hyperparameters.
- We re-implement HiGoC based on CQL and cVAE, and tune over the two hyparameters, learning rate $\in [3e-4, 1e-3]$ and the contribution of KL regularization $\in [0.05, 0.2]$.
- We run Diffuser using the offical repository[9] from the original paper with default hyperparameters.
- We run DD using the offical repository from the original paper with default hyperparameters.

[4] https://github.com/anuragajay/decision-diffuser/tree/main/code.
[5] https://github.com/aviralkumar2907/CQL.
[6] https://github.com/ikostrikov/implicit_q_learning.
[7] https://github.com/sfujim/BCQ
[8] https://github.com/timbmg/VAE-CVAE-MNIST
[9] https://github.com/jannerm/diffuser.

H.1.4. LONG-HORIZON GOAL-REACHING (MULTI-TASK)

For offline RL baselines, we also only evaluated IQL on the multi-task setting. To adapt IQL to the multi-task setting, we borrow the modification from (Janner et al., 2022). Concretely, we modified the Q functions, value function, and policy to be goal-conditioned, that is, the inputs to the model are expanded. For a training sample $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$, we sample goals according to a geometric distribution over the future states

$$\Delta \sim \mathrm{Geom}(1-\gamma) \quad \mathbf{g} = \mathbf{s}_{t+\Delta},$$

recalculated rewards based on the sampled goal similar with hindsight experience replay, and conditioned all goal-conditioned functions on the goal during updating. During testing, we conditioned the policy on the ground-truth goal. We tuned over the same IQL parameters as in the single-task setting.

We implement OPAL based on CQL, and tune over the two hyparameters, learning rate $\in [3e-4, 1e-3]$ and the contribution of KL regularization $\in [0.05, 0.2]$. The other baselines implementation details and hyperparameter settings are the same as for single-task setting.

H.1.5. REWARD-MAXIMIZING WITH SUBOPTIMAL DATA.

- The performance of BC, CQL and IQL in Table 3 is reported in (Kostrikov et al., 2022, Table 1);
- The performance of DT in Table 3 is reported in (Chen et al., 2021, Table 2);
- The performance of TT in Table 3 is reported in (Janner et al., 2021, Table 1);
- The performance of MoReL in Table 3 is reported in (Kidambi et al., 2020, Table 2);
- The performance of Diffuser in Table 3 is reported in the (Janner et al., 2022, Table 2);
- The performance of DD in Table 3 is reported in the (Ajay et al., 2022, Table 1).

H.1.6. REAL-WORLD OFFLINE DECISION-MAKING.

The performance of all baselines in Table 4 is reported in (Qin et al., 2022, Table 1).

H.2. Implementation Details

H.2.1. PLANNING-BASED SUBGOAL EXTRACTION

In the subgoal extraction, the mini-batch k-means++ algorithm uses the clustering method[10] in scikit-learn (Pedregosa et al., 2011) codebase. To ensure the efficient operation of the SoRB algorithm, we limit the number of trajectories in each cluster to about $1,000$ and set the number of cluster centers in conjunction with the size of the dataset. The `max_iter` is set to 200 and `batch_size` is set to 1024. All other hyperparameters follow the default settings.

Regarding the implementation of the SoRB algorithm, we borrowed the official repository[11] given in the original paper, and the hyperparameter settings and tuning range are shown in Table 7.

H.2.2. TRANSFORMER-BASED DIFFUSION MODEL

For the implementation of the tranformer-based diffusion model, we borrowed from the official DiT (Peebles & Xie, 2022) repository[12]. However, since the data processed by HDMI are not images, we use the hyperparameter settings for the transformser in DT[13] to reduce the training cost.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans.

[11] https://colab.research.google.com/github/google-research/google-research/blob/master/sorb/SoRB.ipynb.

[12] https://github.com/facebookresearch/DiT

[13] https://github.com/kzl/decision-transformer

Table 7: Hyperparameter settings and tuning ranges for the proposed offline SoRB.

Phase		Hyperparameter	Value	Tuning Range
Distance Learning	IQN (discrete action)	samples for policy	32	-
		nonlinearity	ReLU	-
		learning rate	3e-4	[1e-4, 1e-3]
	D4PG (continuous action)	learning rate (actor)	5e-5	[1e-5, 1e-4]
		learning rate (critic)	1e-4	[1e-5, 1e-4]
Search on Graph		MaxDist	5	[3, 7]
		search buffer size	1000	-
		discount	1	-
		target update rate	0.95	-
		hidden layer size	1024	[64, 256, 1024]
		number of layers	3	-

H.2.3. Goal and Trajectory Diffuser

In the goal diffuser, we choose the probability p of removing the conditioning information to be 0.25. In the trajectory diffuser, we parameterize the inverse dynamics model F_{θ_I} with a 2-layered MLP with 512 hidden units and `ReLU` nonlinearity. We train the goal diffuser ϵ_{θ_g}, trajectory diffuser ϵ_{θ_s} and inverse dynamics model F_{θ_I} using the Adam optimizer with a learning rate of $2e-4$ and batch size of 32 for $2e6$ training steps. We use $K = 100$ diffusion steps for all diffusers. For different offline decision-making tasks, we use a planning horizon H of 20 in all the D4RL locomotion tasks, 20 in Maze2D and 50 in long-horizon Maze2D, which is much smaller than Diffuser (Janner et al., 2022) and DD (Ajay et al., 2022). We use a guidance scale $s \in \{1.2, 1.4, 1.6, 1.8\}$ but the exact choice varies by task, and we choose context length $C = 20$, which is same as DD (Ajay et al., 2022).

I. Ablation Studies

This section will ablate the design concepts and key modules involved in the HDMI algorithm.

I.1. Independence between subgoals and states

In the graphic model constructed in this paper (shown in Figure 2), the goals are generated independently of the states, which is different from the existing offline RL methods that introduce hierarchical structures in which the goals are generated autoregressively based on the current state(s) (Ajay et al., 2021; Mandlekar et al., 2020; Li et al., 2022).

There is a comparison between autoregressive and simultaneous generation of optimal action sequences which has been discussed in detail in (Janner et al., 2022, §3.1). However, the nature of the problem has changed considerably with the introduction of the hierarchical structure in this paper, so we feel it necessary to elaborate on it here and perform an appropriate ablation analysis.

It is a very natural assumption that goal generation satisfies causality, i.e., the next subgoal is determined based on the past and current state(s). However, decision-making or optimal control can be anti-causal, i.e., the next subgoal can be determined by future information. The above distinction between causality and anticausality can also be considered as the difference between autoregressive and simultaneous generation.

In general reinforcement learning contexts, conditioning on the future emerges from the assumption of future optimality for the purpose of writing a dynamic programming recursion. Concretely, this appears as the future optimality variables $\mathcal{O}_{t:T}$ in the action distribution $\log p\left(\mathbf{a}_t \mid \mathbf{s}_t, \mathcal{O}_{t:T}\right)$ (Levine, 2018). Since the graphic model proposed in this paper is extended on (Levine, 2018), it naturally inherits its conclusions as well.

To more directly compare the performance impact caused by the two generation methods, we compare the performance of HDMI with the recently proposed hierarchical decision transformer, HDT (Correia & Alexandre, 2022) on the long-horizon goal-reaching task, as shown in Table 8. HDT is a hierarchical algorithm for learning a sequence model from demonstrations based on DT (Chen et al., 2021). The high-level mechanism guides the low-level controller through the task by selecting sub-goals for the latter to reach, as shown in Figure 10a. To ensure the fairness of the comparison, we replace the method

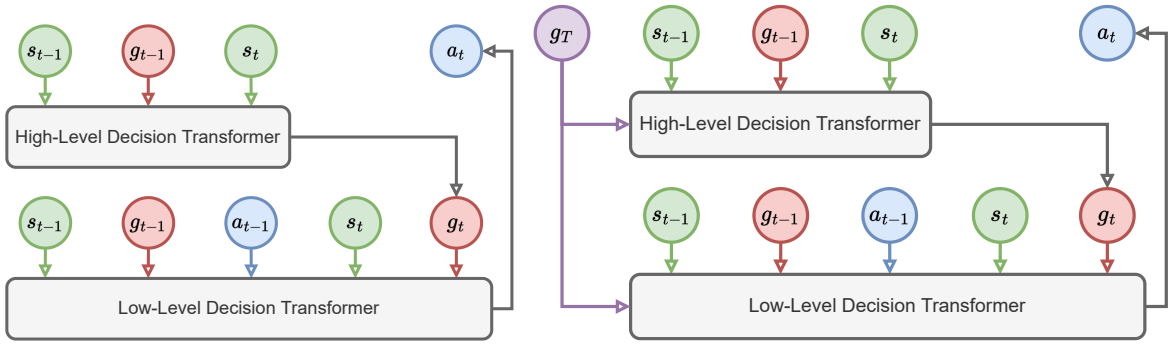


Figure 10: (a) Hierarchical decision transformer (HDT) framework (Correia & Alexandre, 2022). HDT employs two decision transformers (DT) in the form of the high-level DT and low-level DT. The high-level DT guides the low-level DT by selecting the next subgoal, based on the history of subgoals and states, for the low-level DT to try to reach. The low-level DT is conditioned on the history of past states, subgoals and actions to select the next optimal action. (b) Modified HDT framework for goal-reaching taks solving.

of sub-target extraction in HDT with the method adopted in HDMI. In addition, in order to enable HDT to solve the goal-reaching task, we borrowed the idea of TT (Janner et al., 2021) and added the goal state as conditional information to the input of the transformer, as shown in Figure 10b.

Table 8: The performance of HDMI and baselines in Maze2D, a typical long-horizon task with reward sparsity. The Multi2D setting refers to a multi-task variant with episodic, resampled goal locations. Results correspond to the mean and standard error over 5 planning seeds. The suffix number of the environment name indicates that the test map is stitched together from multiple original maps. We emphasize in bold scores within 5 percent of the maximum per task (Kostrikov et al., 2022).

Environment		MPPI	CQL	IQL	OPAL	IRIS	HiGoC	Diffuser	DD	HDT+SoRB	HDMI
Maze2D	U-Maze-3	14.4	3.6	23.2	-	63.8±2.5	61.2±3.3	82.6±1.6	83.9±3.1	82.3±2.4	**103.6±1.7**
Maze2D	Medium-2	5.7	2.3	19.8	-	59.5±4.7	59.8±4.1	87.8±3.1	85.8±3.3	85.2±1.6	**102.1±2.5**
Maze2D	Large-2	3.9	7.7	31.1	-	38.2±1.2	45.4±2.5	87.9±3.8	87.3±1.2	88.5±2.6	**104.7±2.1**
Single-task Average		8.0	6.8	24.7	-	53.8	55.5	86.1	85.7	85.3	**103.5**
Multi2D	U-Maze-3	17.8	-	16.5	-	61.7±3.6	67.9±1.5	85.4±1.8	86.9±3.5	85.6±2.7	**105.4±2.4**
Multi2D	Medium-2	8.1	-	8.9	62.3±2.8	41.4±1.9	52.4±3.7	85.6±3.4	88.2±1.3	85.8±1.5	**104.7±2.3**
Multi2D	Large-2	4.5	-	10.3	55.4±3.7	28.1±3.8	42.1±3.3	89.3±5.8	91.7±2.8	93.6±3.3	**105.8±1.9**
Multi-task Average		10.1	-	11.9	-	43.7	54.1	86.8	88.9	88.3	**105.3**

As can be seen from the table, thanks to the hierarchical structure, HDT shows a similar performance to Diffuser and DD in long-horizon tasks, but there is still a significant gap with HDMI. To further analyze the reasons for this, we visualize the subgoal sequence sampled by HDT and HDMI, as shown in Figure 11. As can be seen from the figure, the subgoal sequence sampled by HDT tends to go around the long way, or even spins in a dead end. This may be because the autoregressive generation lacks the ability of implicit planning, which leads to the tendency to generate suboptimal solutions.

I.2. Importance of Planning-based Subgoal Extraction

The critical challenge of the subgoal extraction is that suboptimal trajectories pollute the dataset, and extracting the corresponding subgoal sequence from each trajectory *independently* will not guarantee subgoal optimality. To this end, we borrow a planning-based online reinforcement learning method, SoRB (Eysenbach et al., 2019), which can automatically find subgoals by providing graphic abstractions of the environment.

To further validate the effectiveness of the modified SoRB in mitigating suboptimal data contamination, we also employed several different subgoal extraction methods, including:

- **Time-sample (Lai et al., 2020) (TS).** Consider the horizon of one trajectory is T, we pick the waypoints on the trajectory with interval of $\frac{T}{k}$ timestep.

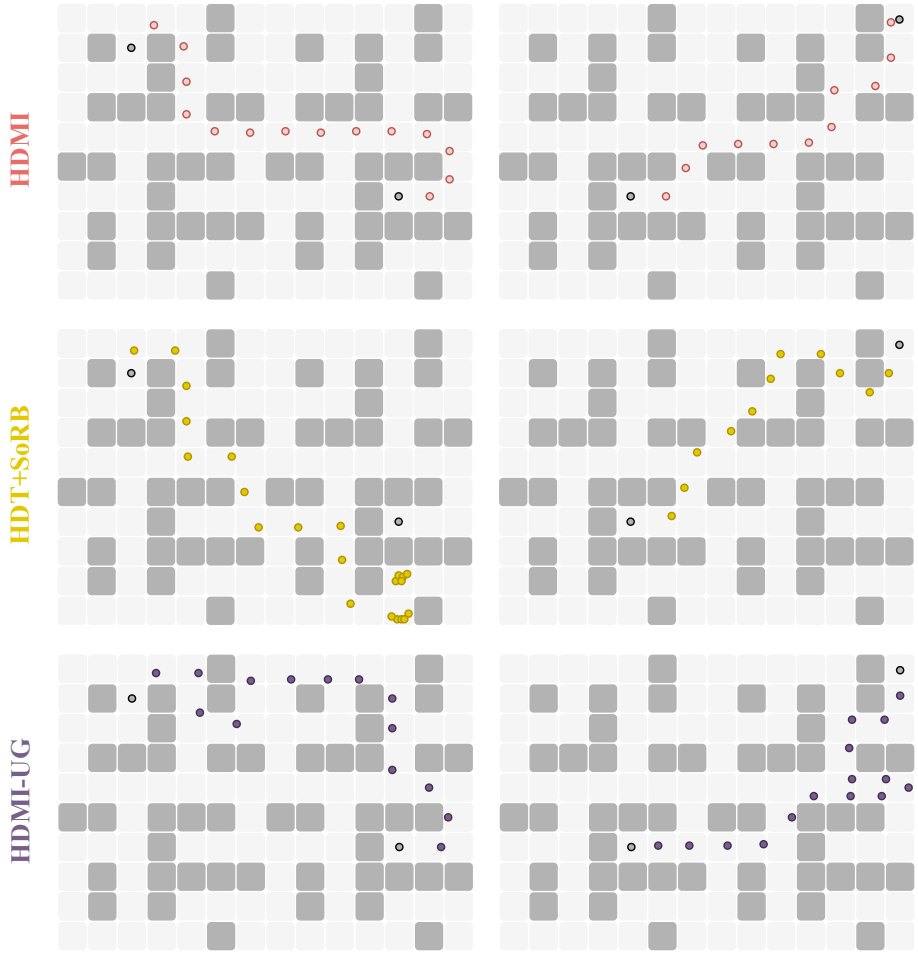


Figure 11: Subgoal sequences sampled by different algorithms in `Large-2` task. Dark gray circles indicate starting or ending points.

- **Route-sample (Lai et al., 2020) (RS).** We denote the distance moved credited to each action a_t as δ_t. Hence, the route length is $\Gamma = \sum_{t=0}^{T-1} \delta_t$. Then, pick the waypoints with interval of $\frac{\Gamma}{k}$ route length.
- **Value-sample (Correia & Alexandre, 2022) (VS).** A subgoal state $s_{g,t}$ for the current state s_t should has high value for the success of the trajectory. Thus we can calculate the returns of all subsequent states $s_{t'}$ in the trajectory starting from the current state s_t, i.e., $W(s_{t'}) = \sum_{k=t+1}^{t'} \frac{r_k}{t'-t}$. After finding the first subgoal, we then iterate over it as the current state to obtain a sequence of subgoals.

It can be seen from Table 9 that different subgoal extraction methods will indeed have a greater impact on the performance of the algorithm. Since TS, RS, and VS are all designed for a single trajectory, pollution caused by suboptimal data cannot be avoided. In the case of a small proportion of suboptimal data, the TS and RS algorithms are closer to the performance of HDMI. However, as the pollution situation continues to increase, the performance of both has also declined significantly. Compared with TS and RS, the VS algorithm is much inferior. We speculate that the reason is that there is greater uncertainty in the reward signal, combined with suboptimal data pollution, making it more difficult to model sub-goal sequences extracted from VS.

In addition to utilizing the SoRB for generating training data, it is possible to apply SoRB directly during the testing phase to produce subgoals without any additional training of the goal diffuser. This ablation algorithm is denoted as HDMI-S. It must be noted that since SoRB is an online RL technique that requires maintaining a search replay buffer before generating subgoals, it cannot be utilized for solving offline tasks directly. In the online setting, the agent collects the buffer data during the exploration stage. However, only the offline dataset can be relied upon under the offline setting.

Table 9: The performance of HDMI and baselines in D4RL, a standardize long-horizon reward-maximizing environment, in terms of normalized average returns. Results for DD and HDMI correspond to the mean and standard error over 5 planning seeds. We emphasize in bold scores within 5 percent of the maximum per task (Kostrikov et al., 2022).

Dataset	Environment	BC	CQL	IQL	DT	TT	MoReL
Med-Expert	HalfCheetah	55.2	91.6	86.7	86.8	**95**	53.3
Med-Expert	Hopper	52.5	105.4	91.5	107.6	110.0	108.7
Med-Expert	Walker2d	**107.5**	**108.8**	**109.6**	**108.1**	101.9	95.6
Medium	HalfCheetah	42.6	44.0	47.4	42.6	46.9	42.1
Medium	Hopper	52.9	58.5	66.3	67.6	61.1	**95.4**
Medium	Walker2d	75.3	72.5	78.3	74.0	79	77.8
Med-Replay	HalfCheetah	36.6	**45.5**	**44.2**	36.6	41.9	40.2
Med-Replay	Hopper	18.1	95	94.7	82.7	91.5	93.6
Med-Replay	Walker2d	26.0	77.2	73.9	66.6	**82.6**	49.8
Average		51.9	77.6	77	74.7	78.9	72.9
Dataset	**Environment**	**Diffuser**	**DD**	**HDMI-TS**	**HDMI-RS**	**HDMI-VS**	**HDMI**
Med-Expert	HalfCheetah	79.8	90.6±1.3	86.7±1.5	86.3±1.1	87.5±1.8	92.1±1.4
Med-Expert	Hopper	107.2	111.8±1.8	108.9±0.7	107.2±1.2	106.7±2.2	**113.5±0.9**
Med-Expert	Walker2d	**108.4**	**108.8±1.7**	105.1±1.3	105.8±0.8	103.2±0.9	**107.9±1.2**
Medium	HalfCheetah	44.2	**49.1±1.0**	42.2±0.8	41.3±0.7	40.4±2.1	**48.0±0.9**
Medium	Hopper	58.5	79.3±3.6	61.6±1.8	62.4±1.5	60.5±1.5	76.4±2.6
Medium	Walker2d	79.7	**82.5±1.4**	74.5±1.2	74.8±0.8	73.0±1.9	79.9±1.8
Med-Replay	HalfCheetah	42.2	39.3±4.1	38.0±0.6	37.6±0.6	36.8±0.5	**44.9±2.0**
Med-Replay	Hopper	96.8	**100±0.7**	84.9±0.7	82.8±1.3	82.7±1.3	**99.6±1.5**
Med-Replay	Walker2d	61.2	75±4.3	65.2±2.4	66.8±1.3	62.6±1.7	**80.7±2.1**
Average		75.3	81.8	74.1	73.9	72.6	**82.6**

It has been previously highlighted in the primary text that searching using the entire offline dataset is excessively time-consuming. Thus, we have also implemented the clustering method to extract trajectories with start states and goal states similar to the test task from the offline dataset and utilize these trajectories to construct the search replay buffer. The AntMaze dataset is used to validate the performance of HDMI-S vs. HDMI, as depicted in Table 10.

Table 10: The performance in AntMaze, a typical long-horizon task with reward sparsity. MultiAnt-Diverse is a multi-task variant of AntMaze-Diverse with episodic, resampled goal locations. Results correspond to the mean and standard error over 5 planning seeds. The suffix number of the environment name indicates that the test map is stitched together from multiple original maps. HDMI-S indicates that the SoRB algorithm is directly used to generate subgoals during the test phase, and the numbers in parentheses represent the multiples of time consumed when inferring relative to the HDMI.

Environment		HDMI	HDMI-S
AntMaze-Play	U-Maze-3	**86.1 ± 2.4 (1x)**	82.5 ± 1.6 (2.2x)
AntMaze-Diverse	U-Maze-3	**73.7 ± 1.1 (1x)**	66.2 ± 2.1 (2.2x)
AntMaze-Diverse	Large-2	**71.5 ± 3.5 (1x)**	60.5 ± 1.9 (3.6x)
Single-task Average		**77.1 (1x)**	70.7 (2.7x)
MultiAnt-Diverse	Large-2	**73.6 ± 3.8 (1x)**	60.9 ± 2.3 (3.4x)
Multi-task Average		**73.6 (1x)**	60.9 (3.4x)

The table reveals that HDMI-S takes longer to execute and exhibits inferior performance compared to HDMI. We attribute this disparity to the susceptibility of the search replay buffer to the clustering quality.

Furthermore, in the test phase, substituting the goal diffuser with SoRB renders the HDMI-S unable to tackle the reward-maximizing task, as SoRB necessitates prior knowledge of the goal state at runtime. In contrast, the HDMI can employ more generalized conditional information (not restricted to the goal state) for subgoal generation due to "algorithm distillation" from SoRB during the training phase utilizing the conditional generation model. For example, the expected cumulative reward serves as conditional information in the reward-maximizing task. In summary, it is an intriguing research avenue to enhance the integration of planning methods with conditional generative models. We intend to delve deeper into this area in our future work.

I.3. Importance of Transformer-based Diffusion

Existing work has shown that the capture of correlations between elements is critical to the success of diffusion models. Different from related works that use the diffusion model to denoise the entire state trajectory or state-action trajectory (Janner et al., 2022; Ajay et al., 2022), in the goal diffusion procedure of the upper level, we denoise the sparser subgoal trajectory. The long-range dependence between subgoals makes the U-Net-like structure based on local convolution no longer the optimal choice. This motivates us to use the transformer as the skeleton of the diffusion model instead of the commonly used U-Net-like structure.

In order to bolster the efficacy of the transformer-based diffusion model in capturing sub-goal sequences, we have conducted an experiment in which we substituted the skeletal components of both the goal diffuser and the trajectory diffuser with U-Net structures. We have also devised two distinct ablation techniques, HDMI-UG (i.e., Goal diffuser with U-Net) and HDMI-UT ((i.e., Trajectory diffuser with U-Net), to evaluate the performance of the modified model. Our findings shed light on the superiority of the transformer-based approach in this context, as demonstrated through the results of our experiments.

Concretely, we represent the noise model ϵ_θ with a temporal U-Net and borrow the code from (Janner et al., 2022)[14]. The temporal U-Net consists of a U-Net structure with 6 repeated residual blocks. Each block consists of two temporal convolutions, each followed by group norm (Wu & He, 2018). The Mish nonlinearity (Misra, 2020) is added before the final output. Timestep and condition embeddings are 128-dimensional vectors which are produced by separate 2-layered MLP. Each layer consists of 256 hidden units and Mish nonlinearity. The timestep and condition embeddings are then concatenated together before getting added to the activations of the first temporal convolution within each block.

Table 11: The performance of HDMI and baselines in Maze2D, a typical long-horizon task with reward sparsity. The Multi2D setting refers to a multi-task variant with episodic, resampled goal locations. Results correspond to the mean and standard error over 5 planning seeds. The suffix number of the environment name indicates that the test map is stitched together from multiple original maps. We emphasize in bold scores within 5 percent of the maximum per task (Kostrikov et al., 2022).

Environment		MPPI	CQL	IQL	OPAL	IRIS	HiGoC	Diffuser	DD	HDMI-UG	HDMI-UT	HDMI
Maze2D	U-Maze-3	14.4	3.6	23.2	-	63.8±2.5	61.2±3.3	82.6±1.6	83.9±3.1	82.6±2.1	**103.1±1.5**	**103.6±1.7**
Maze2D	Medium-2	5.7	2.3	19.8	-	59.5±4.7	59.8±4.1	87.8±3.1	85.8±3.3	88.9±1.9	**101.8±1.7**	**102.1±2.5**
Maze2D	Large-2	3.9	7.7	31.1	-	38.2±1.2	45.4±2.5	87.9±3.8	87.3±1.2	94.3±1.3	**104.9±1.7**	**104.7±2.1**
Single-task Average		8.0	6.8	24.7	-	53.8	55.5	86.1	85.7	88.6	**103.3**	**103.5**
Multi2D	U-Maze-3	17.8	-	16.5	-	61.7±3.6	67.9±1.5	85.4±1.8	86.9±3.5	83.8±1.7	**105.8±2.2**	**105.4±2.4**
Multi2D	Medium-2	8.1	-	8.9	62.3±2.8	41.4±1.9	52.4±3.7	85.6±3.4	88.2±1.3	90.5±1.1	**103.5±2.8**	**104.7±2.3**
Multi2D	Large-2	4.5	-	10.3	55.4±3.7	28.1±3.8	42.1±3.3	89.3±5.8	91.7±2.8	95.6±2.1	**105.2±1.2**	**105.8±1.9**
Multi-task Average		10.1	-	11.9	-	43.7	54.1	86.8	88.9	90.0	104.8	**105.3**

Table 11 presents two main observations. Firstly, the utilization of either U-Net (HDMI-UT) or transformer (HDMI) for lower-level trajectory diffusers has a negligible effect on performance, with the transformer exhibiting slightly superior performance than U-Net. Secondly, the transformer model (HDMI) considerably outperforms U-Net (HDMI-UG) in modeling subgoal trajectories, particularly when dealing with sparser subgoals (i.e., smaller maps), thus resulting in a wider performance discrepancy between the two models. We also present a visual depiction of the subgoal sequence produced by transformer-(HDMI) and U-Net-based (HDMI-UG) goal diffusers, as illustrated in Figure 11. The figure indicates that the subgoal trajectory generated by HDMI-UG tends to take longer routes, possibly due to U-Net-based diffusion emphasizing local coherence while overlooking global constraints, hence making it more prone to suboptimal solutions.

J. Limitations

This section summarizes the limitations and future works of HDMI for model training as well as solvable problems.

J.1. Joint training of goal and trajectory diffusers

In the cascade trajectory diffusion process, the goal-conditional policy learning is translated into an inpainting problem by replacing the sampled states with conditioning subgoals. However, the goal diffuser uses more information (i.e., extrinsic rewards) than the trajectory diffuser, making the discovered subgoals potentially infeasible. In other words, the training

[14] https://github.com/jannerm/diffuser.

process of the goal diffuser is separated from the trajectory diffuser.

One possible solution is to borrow the adaptive guiding from guided policy search (GPS) (Levine & Koltun, 2013), and introduce an extra conditional information for the goal diffuser to encourage the subgoals distribution to be consistent with the trajectory distribution. Specifically, we can redefine the conditional information of the goal diffuser as

$$\bar{y}(\tau_g) := \exp\Big(\sum_{t=0}^{T-1} r(s_t, a_t) + \sum_{i=1}^{N} \log p(\tau_{sa}^i; y(\tau_{sa}^i))\Big)$$

where $\log p(\tau_{sa}^i; y(\tau_{sa}^i))$ represents the log-likelihood of the sub-trajectory generated by the trajectory diffuser conditioned on the subgoal g_i.

The equivalence between diffusion models and score-matching (Song et al., 2021) enables exact log-likelihood computation for diffusion models. Specifically, we can leverage the instantaneous change-of-variable formula (Chen et al. (2018, Theorem 1) and Grathwohl et al. (2019, Equation (4))) to compute the unknown data density p_0 from the known prior density p_T with numerical ODE solvers (Song, 2021). Therefore, we can directly use the trick in GPS. We plan to leave it for follow-up work to explore in depth.

J.2. Translating goals into skills

The subgoal extraction approach HDMI used are unsuited to find subtasks without definite subgoals associated with them. Consider an example of a subtask "drive through traffic," which is a part of a longer horizon task of reaching a destination in the autonomous driving scenario. This subtask requires an agent to maneuver a vehicle around traffic smoothly without any particular subgoal in the state space. Therefore, a general subgoal extraction approach is desired that can directly discover a set of diverse *skills*[15], instead of learning them through subgoals. In existing works, the skill is generally obtained by encoding the trajectory data and is represented as a low-dimensional latent vector (Lynch et al., 2020; Pertsch et al., 2021a; Ajay et al., 2021; Rosete-Beas et al., 2022; Lee et al., 2022b).

In addition, the subgoal extraction and the subsequent training of diffusion models are also independent of each other in this paper. An ideal way is that the trajectory diffuser can automatically generate optimal skills during training and generate optimal action sequences based on the skills. Fortunately, recent works studying diffusion models begin to focus on performing the diffusion in latent space, rather than observation space, as done in (Vahdat et al., 2021; Rombach et al., 2022; Zhou et al., 2022). This makes it possible to generate action sequences end-to-end based on the dataset alone, which we also plan to leave for further exploration in subsequent work. Incidentally, performing the diffusion in the latent space also enables HDMI to be extended to image-based offline decision-making.

[15] A skill refers to "the policy of a subtask in the sense that it semantically represents the ability to do something well" (Pateria et al., 2021).